A Project Report on

Netflix Database Management System

Department of Information Technology Faculty of Technology,
Dharmsinh Desai University College Road, Nadiad-387001
October-2022

Developed By:

1.) Patel Jenish-IT108
2.) Patel Megh-IT109
3.) Patel Het-IT106

Guided By

Prof. Archana N. Vyas
& Prof. Shweta Jambukia

CERTIFICATE

THIS CERTIFICATE IS TO CERTIFY THAT THE PROJECT ENTITLED “NETFLIX DATABASE MANAGEMENT SYSTEM” IS , A BONAFIDE REPORT OF THE WORK CARRIED OUT BY,

1) Patel Jenish	Student ID no:20ITUBS039
2) Patel Megh	Student ID no:20ITUON146
3)Patel Het	Student ID no:20ITUON154

OF DEPARTMENT OF INFORMATION TECHNOLOGY, SEMESTER V, UNDER THE GUIDANCE AND SUPERVISION FOR THE SUBJECT DATABASE MANAGEMENT SYSTEM. THEY WERE INVOLVED IN PROJECT TRAINING DURING THE ACADEMIC YEAR 2022-2023.

DHARAMSINH DESAI UNIVERSITY NADIAD

COMMENDATION

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project "Netflix DataBase Management System ".

The success and ultimate conclusion of this project necessitated a great deal of advice and support from a large number of individuals and we are incredibly fortunate to have received it all along with the project's completion.

We owe a debt of appreciation to **Prof. Archana N. Vyas & Prof. Shweta Jambukia**, our project guide, who took an interest in our project work and directed us through it till it was completed by giving all of the required assistance for creating a solid Database System.

We'd also want to express our gratitude to all of our speakers. Finally, we express our gratitude to all of our friends and colleagues.

INDEX

1.SYSTEM OVERVIEW

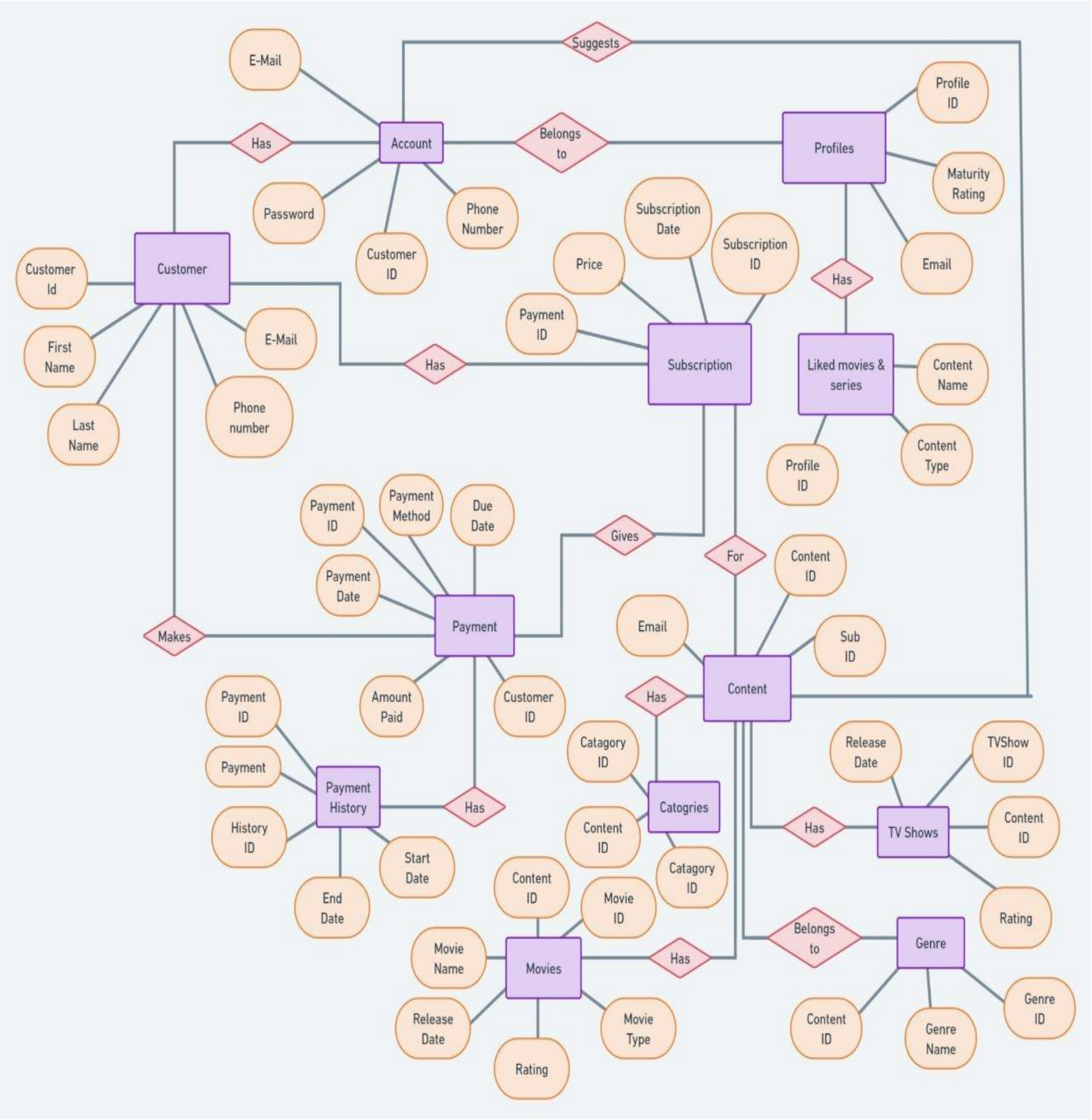
1.1) Current System & Objectives

Our database will be designed based on OTT Streaming service ‘NETFLIX’ database management system. Our major focus will be on structuring the data such a way that a customer of a Streaming service and easily access and use our database, And we can easily manage all the data of customers and our content. It will also be done in an effective manner since we don’t want our Database to become redundant.So, We will try to make the database with the highest power possible.

1.2) Advantage of System

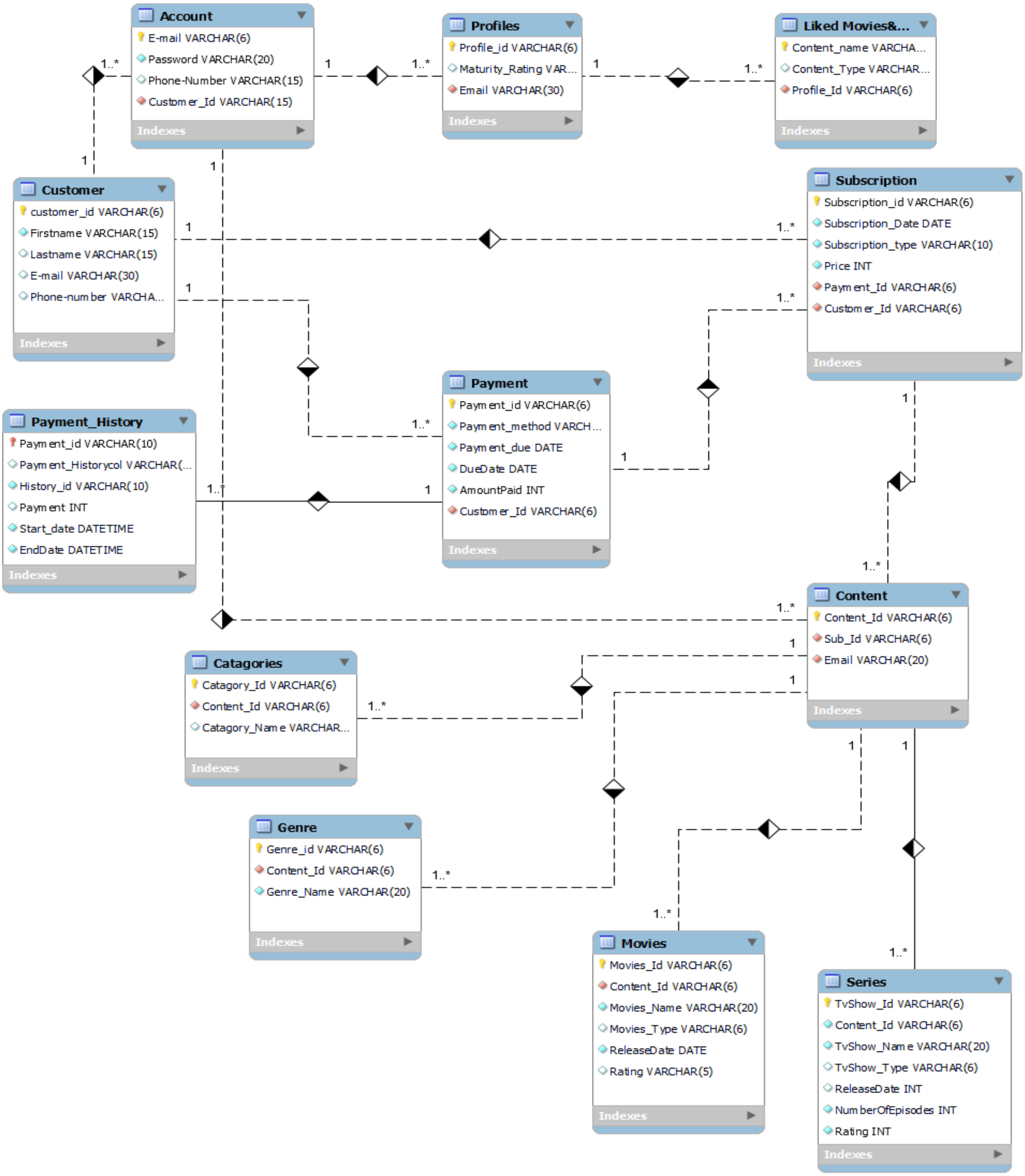
Netflix is Server and Client-side Application. All customer who has subscription can Watch Movies & Tv-shows, according to their plan they have a different video quality and allowed number of connected devices. Both customer and their payment data and movies and tv-shows which customers are streaming everything is managed very accurately. This type of Database System allows Netflix to suggest movies & Tvshows of their like or of type match of what movies they liked in the past.

2) ER-Entity Relational Diagram



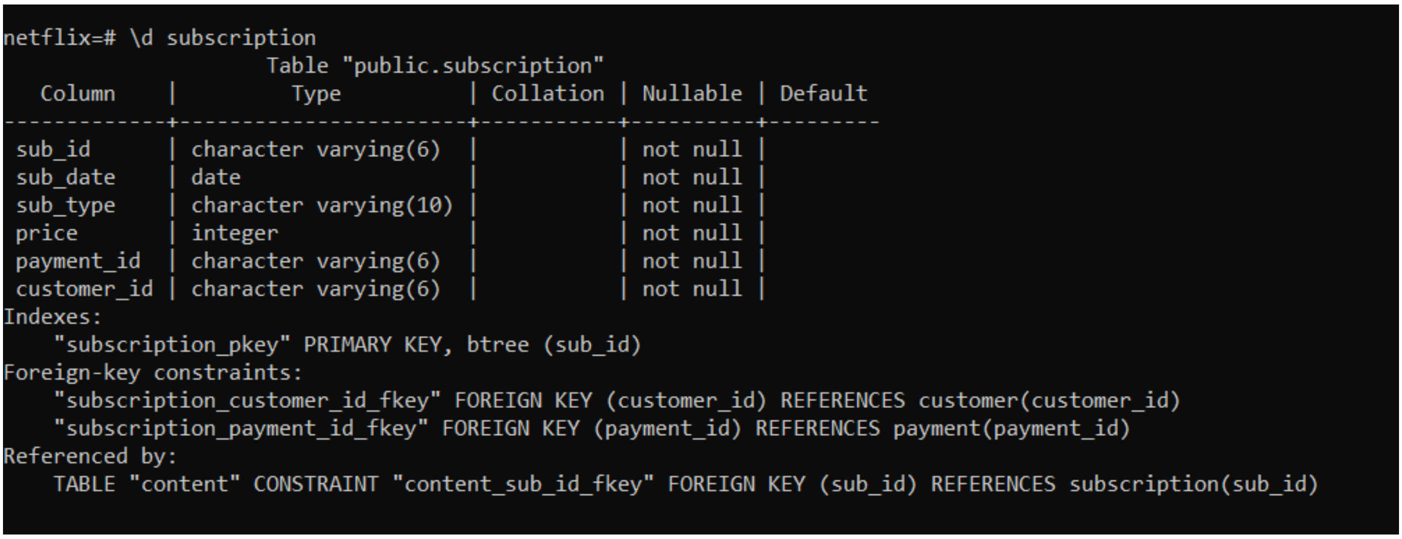
Link => https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD
Password: 12345

3. RELATIONAL SCHEMA



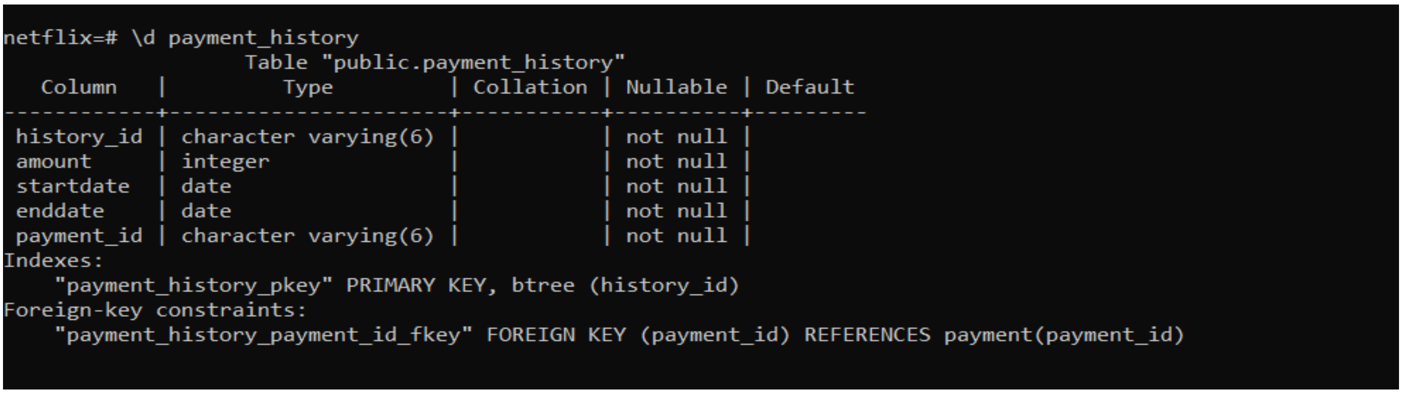
4. Data Dictionary

4.1) Subscription



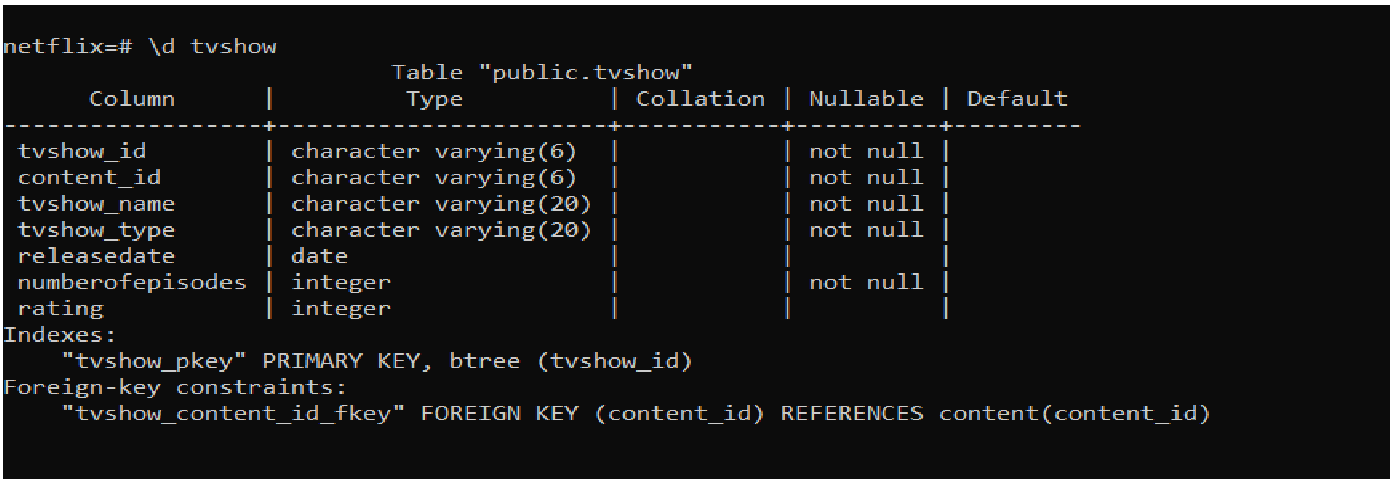
```
netflix=# \d subscription
                     Table "public.subscription"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 sub_id      | character varying(6)  |           | not null |
 sub_date    | date                  |           | not null |
 sub_type    | character varying(10) |           | not null |
 price       | integer               |           | not null |
 payment_id  | character varying(6)  |           | not null |
 customer_id | character varying(6)  |           | not null |
Indexes:
    "subscription_pkey" PRIMARY KEY, btree (sub_id)
Foreign-key constraints:
    "subscription_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
    "subscription_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
Referenced by:
    TABLE "content" CONSTRAINT "content_sub_id_fkey" FOREIGN KEY (sub_id) REFERENCES subscription(sub_id)
```

4.2) Payment History



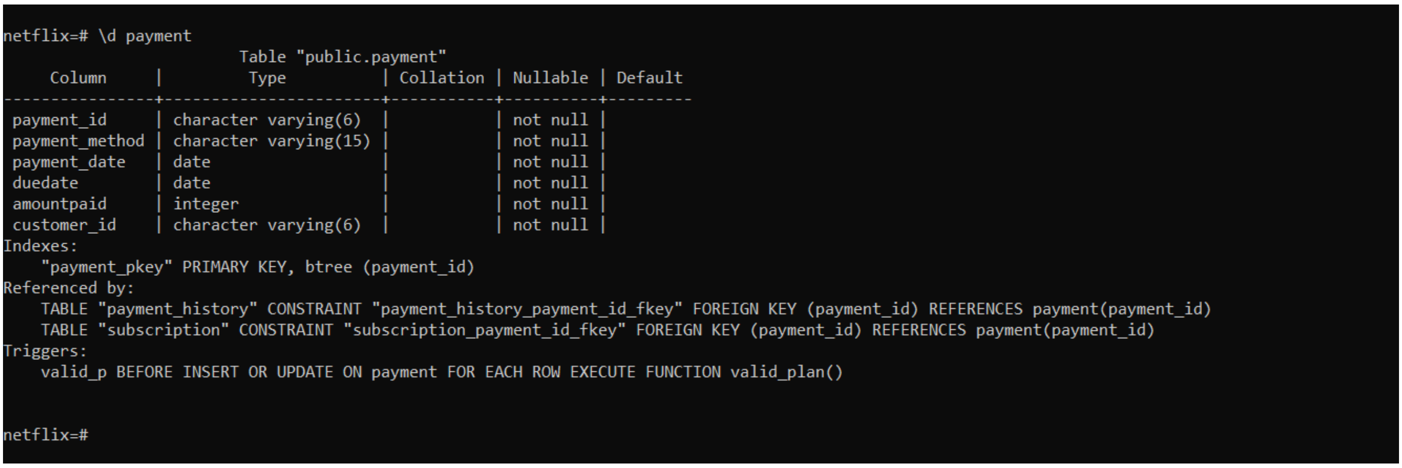
```
netflix=# \d payment_history
                  Table "public.payment_history"
   Column   |         Type         | Collation | Nullable | Default
------------+----------------------+-----------+----------+---------
 history_id | character varying(6) |           | not null |
 amount     | integer              |           | not null |
 startdate  | date                 |           | not null |
 enddate    | date                 |           | not null |
 payment_id | character varying(6) |           | not null |
Indexes:
    "payment_history_pkey" PRIMARY KEY, btree (history_id)
Foreign-key constraints:
    "payment_history_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
```

4.3) TV-Show



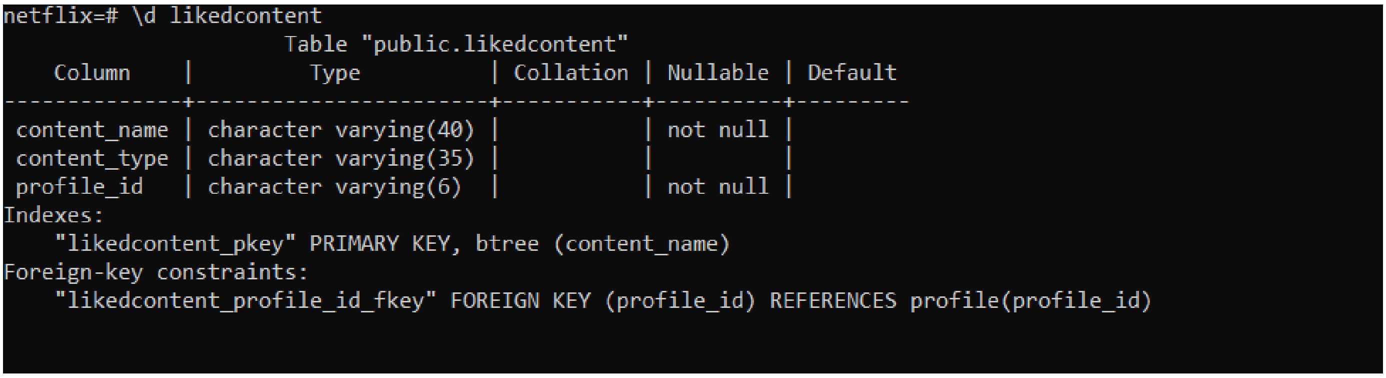
```
netflix=# \d tvshow
                           Table "public.tvshow"
      Column       |          Type          | Collation | Nullable | Default
-------------------+------------------------+-----------+----------+---------
 tvshow_id         | character varying(6)   |           | not null |
 content_id        | character varying(6)   |           | not null |
 tvshow_name       | character varying(20)  |           | not null |
 tvshow_type       | character varying(20)  |           | not null |
 releasedate       | date                   |           |          |
 numberofepisodes  | integer                |           | not null |
 rating            | integer                |           |          |
Indexes:
    "tvshow_pkey" PRIMARY KEY, btree (tvshow_id)
Foreign-key constraints:
    "tvshow_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
```

4.4) Payment



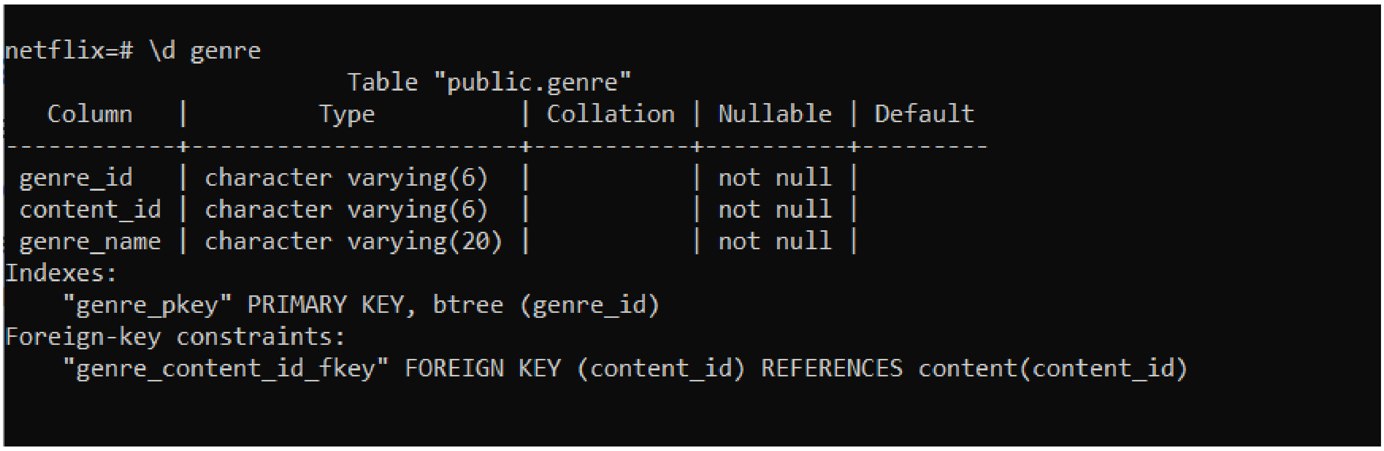
```
netflix=# \d payment
                        Table "public.payment"
     Column     |         Type          | Collation | Nullable | Default
----------------+-----------------------+-----------+----------+---------
 payment_id     | character varying(6)  |           | not null |
 payment_method | character varying(15) |           | not null |
 payment_date   | date                  |           | not null |
 duedate        | date                  |           | not null |
 amountpaid     | integer               |           | not null |
 customer_id    | character varying(6)  |           | not null |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (payment_id)
Referenced by:
    TABLE "payment_history" CONSTRAINT "payment_history_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
    TABLE "subscription" CONSTRAINT "subscription_payment_id_fkey" FOREIGN KEY (payment_id) REFERENCES payment(payment_id)
Triggers:
    valid_p BEFORE INSERT OR UPDATE ON payment FOR EACH ROW EXECUTE FUNCTION valid_plan()


netflix=#
```

4.5) Liked Content



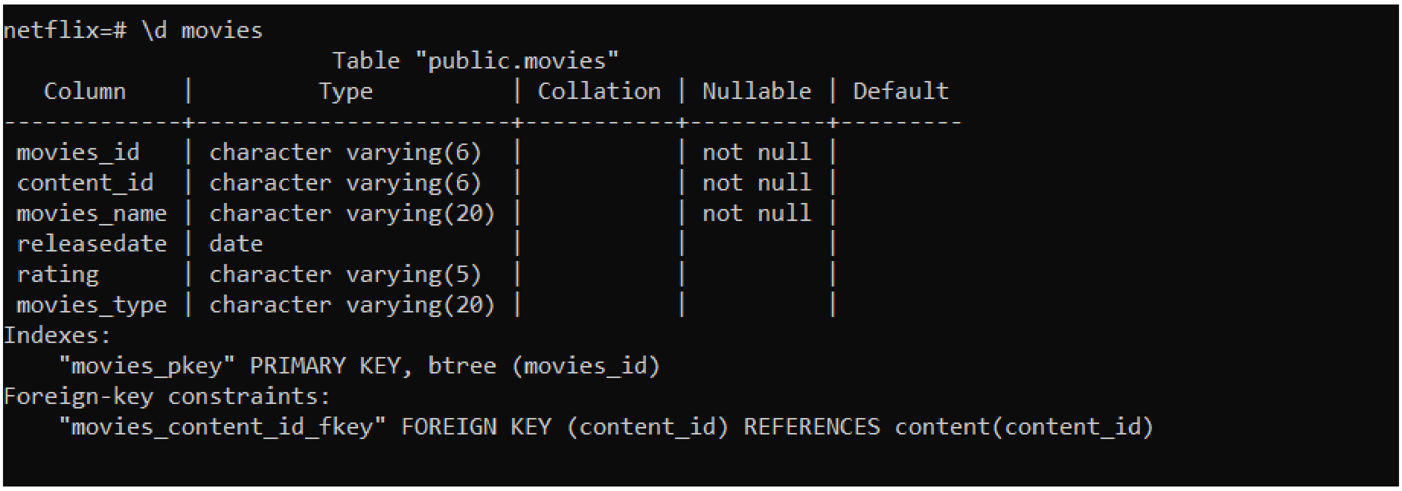
```
netflix=# \d likedcontent
                     Table "public.likedcontent"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 content_name | character varying(40) |           | not null |
 content_type | character varying(35) |           |          |
 profile_id   | character varying(6)  |           | not null |
Indexes:
    "likedcontent_pkey" PRIMARY KEY, btree (content_name)
Foreign-key constraints:
    "likedcontent_profile_id_fkey" FOREIGN KEY (profile_id) REFERENCES profile(profile_id)
```

4.6) Genre



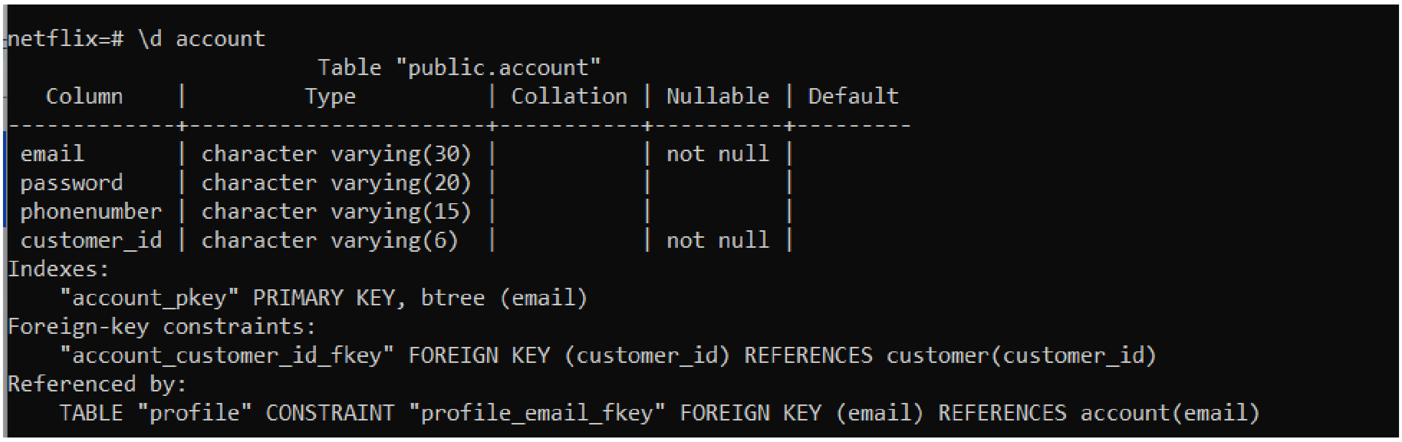
```
netflix=# \d genre
                        Table "public.genre"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 genre_id   | character varying(6)  |           | not null |
 content_id | character varying(6)  |           | not null |
 genre_name | character varying(20) |           | not null |
Indexes:
    "genre_pkey" PRIMARY KEY, btree (genre_id)
Foreign-key constraints:
    "genre_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
```

4.7) Movies



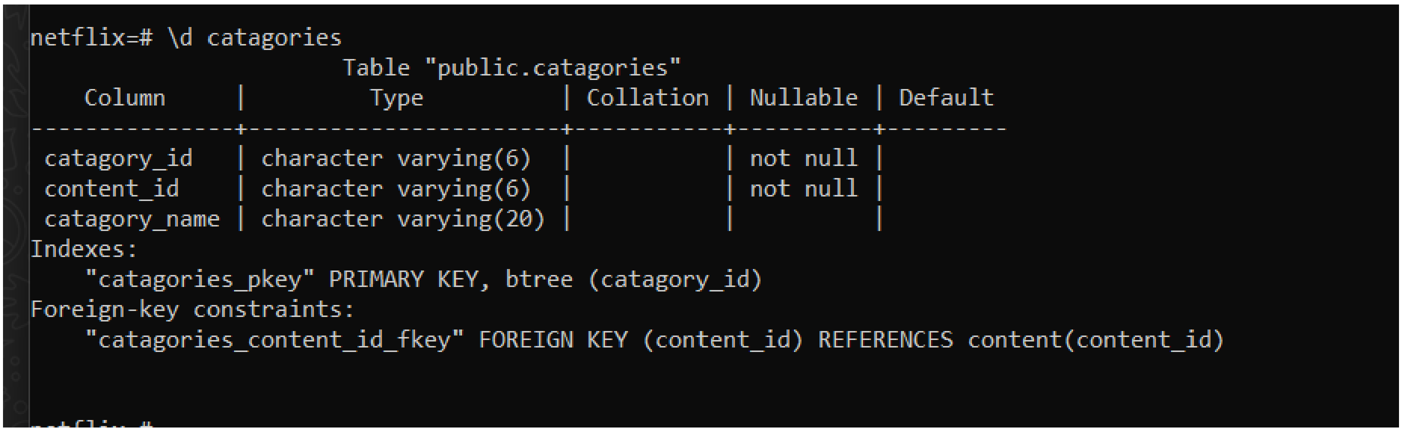
```
netflix=# \d movies
                        Table "public.movies"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 movies_id   | character varying(6)  |           | not null |
 content_id  | character varying(6)  |           | not null |
 movies_name | character varying(20) |           | not null |
 releasedate | date                  |           |          |
 rating      | character varying(5)  |           |          |
 movies_type | character varying(20) |           |          |
Indexes:
    "movies_pkey" PRIMARY KEY, btree (movies_id)
Foreign-key constraints:
    "movies_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
```

4.8) Account



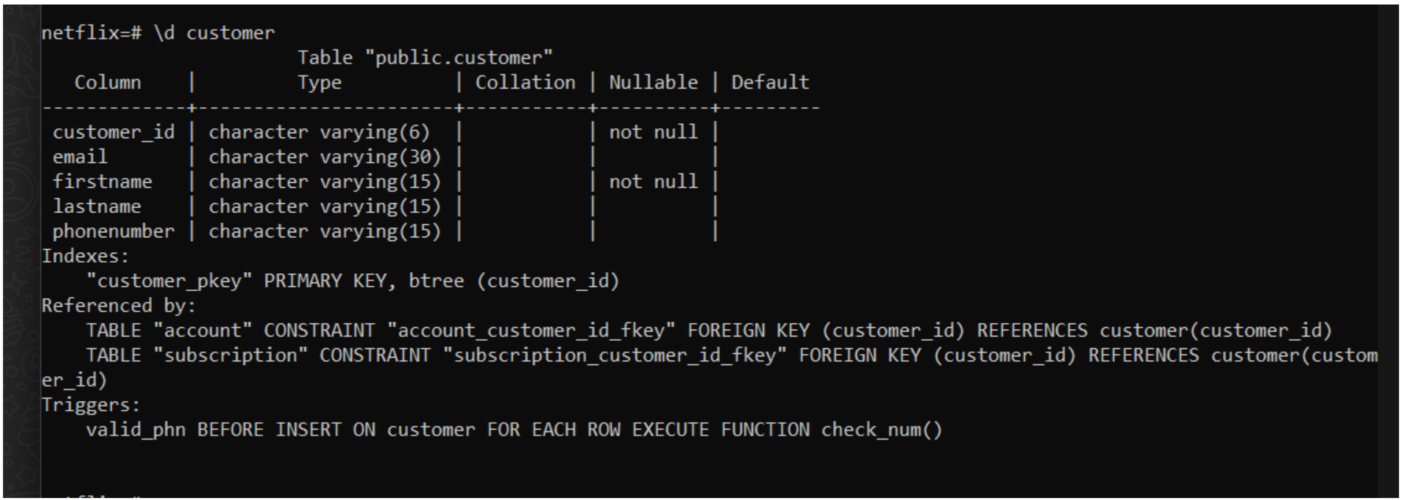
```
netflix=# \d account
                       Table "public.account"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 email       | character varying(30) |           | not null |
 password    | character varying(20) |           |          |
 phonenumber | character varying(15) |           |          |
 customer_id | character varying(6)  |           | not null |
Indexes:
    "account_pkey" PRIMARY KEY, btree (email)
Foreign-key constraints:
    "account_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
Referenced by:
    TABLE "profile" CONSTRAINT "profile_email_fkey" FOREIGN KEY (email) REFERENCES account(email)
```

4.9) Categories



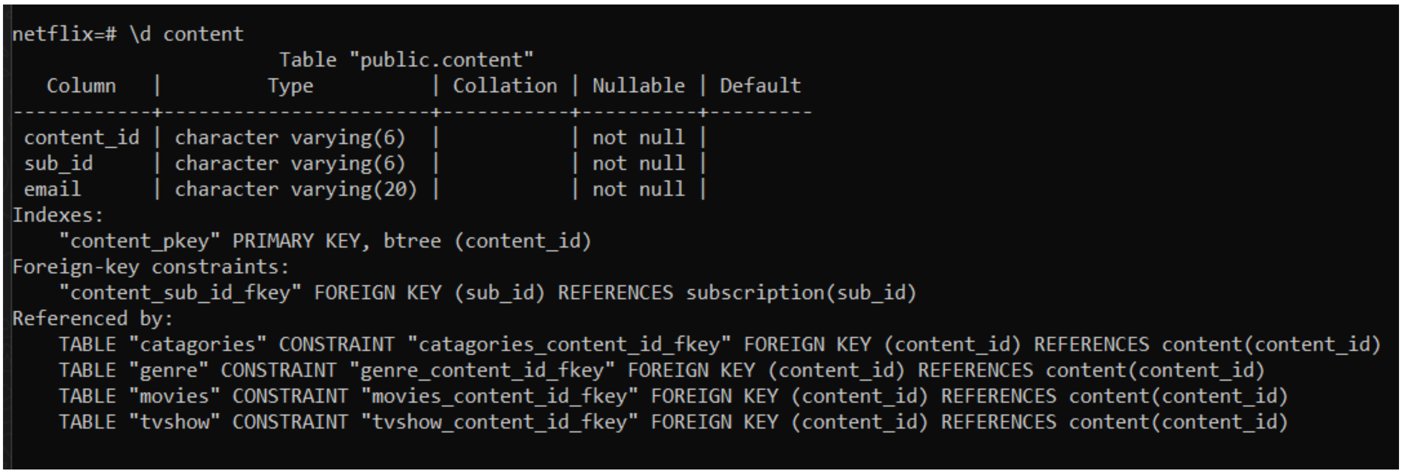
```
netflix=# \d catagories
                       Table "public.catagories"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 catagory_id   | character varying(6)  |           | not null |
 content_id    | character varying(6)  |           | not null |
 catagory_name | character varying(20) |           |          |
Indexes:
    "catagories_pkey" PRIMARY KEY, btree (catagory_id)
Foreign-key constraints:
    "catagories_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
```

4.10) Customer



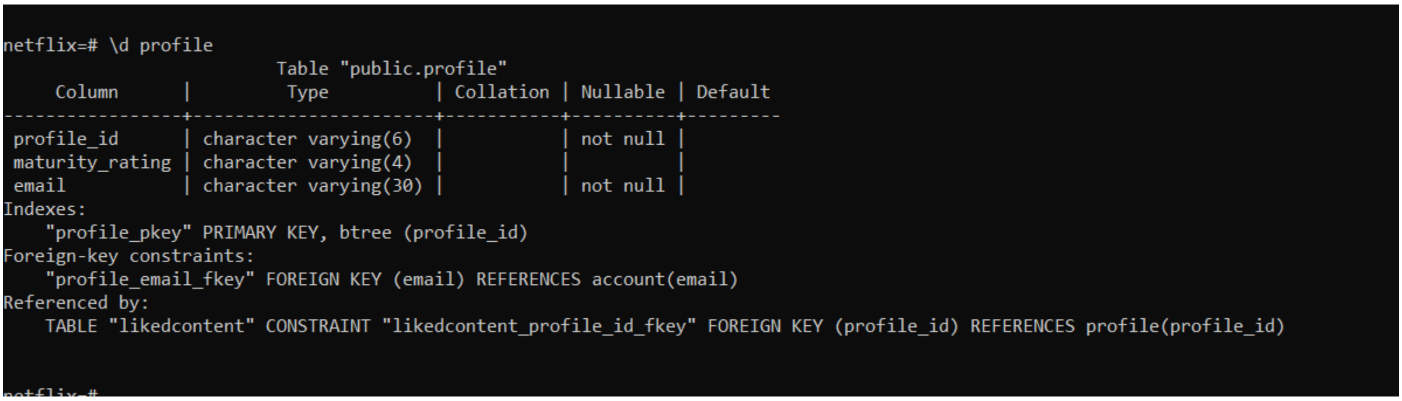
```
netflix=# \d customer
                       Table "public.customer"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 customer_id | character varying(6)  |           | not null |
 email       | character varying(30) |           |          |
 firstname   | character varying(15) |           | not null |
 lastname    | character varying(15) |           |          |
 phonenumber | character varying(15) |           |          |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (customer_id)
Referenced by:
    TABLE "account" CONSTRAINT "account_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
    TABLE "subscription" CONSTRAINT "subscription_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(custom
er_id)
Triggers:
    valid_phn BEFORE INSERT ON customer FOR EACH ROW EXECUTE FUNCTION check_num()
```

4.11) content



```
netflix=# \d content
                       Table "public.content"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 content_id | character varying(6)  |           | not null |
 sub_id     | character varying(6)  |           | not null |
 email      | character varying(20) |           | not null |
Indexes:
    "content_pkey" PRIMARY KEY, btree (content_id)
Foreign-key constraints:
    "content_sub_id_fkey" FOREIGN KEY (sub_id) REFERENCES subscription(sub_id)
Referenced by:
    TABLE "catagories" CONSTRAINT "catagories_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
    TABLE "genre" CONSTRAINT "genre_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
    TABLE "movies" CONSTRAINT "movies_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
    TABLE "tvshow" CONSTRAINT "tvshow_content_id_fkey" FOREIGN KEY (content_id) REFERENCES content(content_id)
```

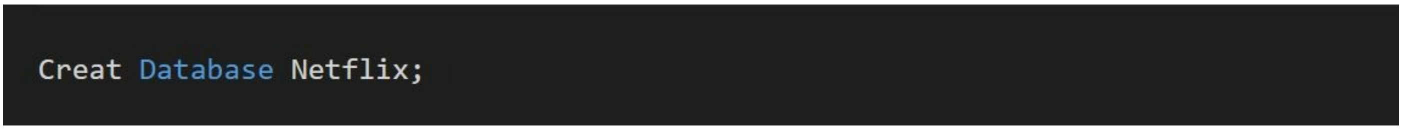
4.12) Profile



```
netflix=# \d profile
                         Table "public.profile"
     Column      |         Type          | Collation | Nullable | Default
-----------------+-----------------------+-----------+----------+---------
 profile_id      | character varying(6)  |           | not null |
 maturity_rating | character varying(4)  |           |          |
 email           | character varying(30) |           | not null |
Indexes:
    "profile_pkey" PRIMARY KEY, btree (profile_id)
Foreign-key constraints:
    "profile_email_fkey" FOREIGN KEY (email) REFERENCES account(email)
Referenced by:
    TABLE "likedcontent" CONSTRAINT "likedcontent_profile_id_fkey" FOREIGN KEY (profile_id) REFERENCES profile(profile_id)


netflix=#
```

5. Data Implementation

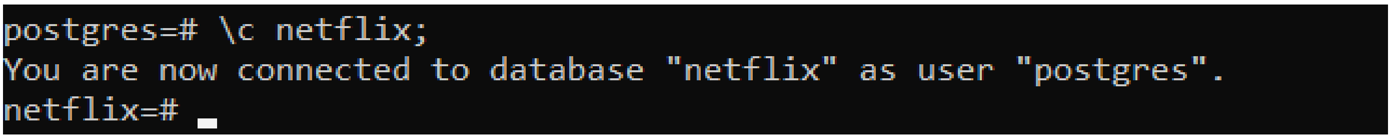
➔ **First Create Database name ->Netflix**



```
Creat Database Netflix;
```

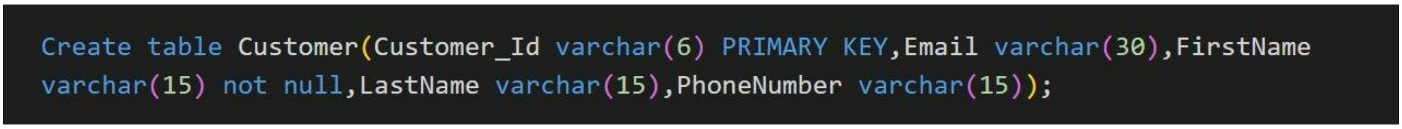
➔ **Schema => Total 12 Table**

➔ **Connecting To The ‘Netflix’ Database**



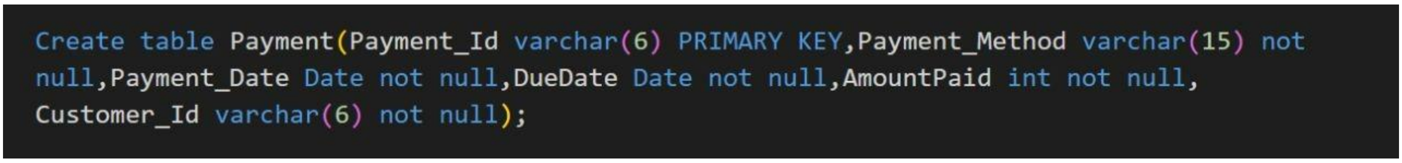
```
postgres=# \c netflix;
You are now connected to database "netflix" as user "postgres".
netflix=# _
```

1) Customer



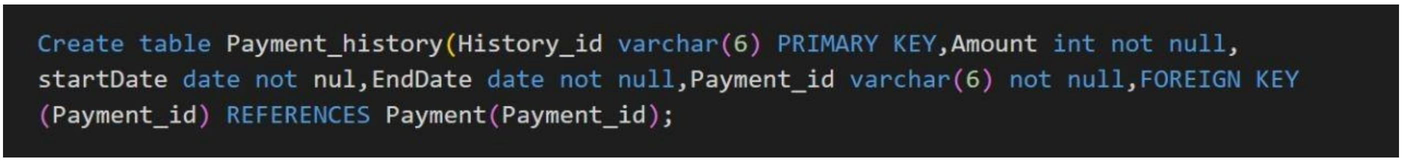
```
Create table Customer(Customer_Id varchar(6) PRIMARY KEY,Email varchar(30),FirstName
varchar(15) not null,LastName varchar(15),PhoneNumber varchar(15));
```

2) Payment



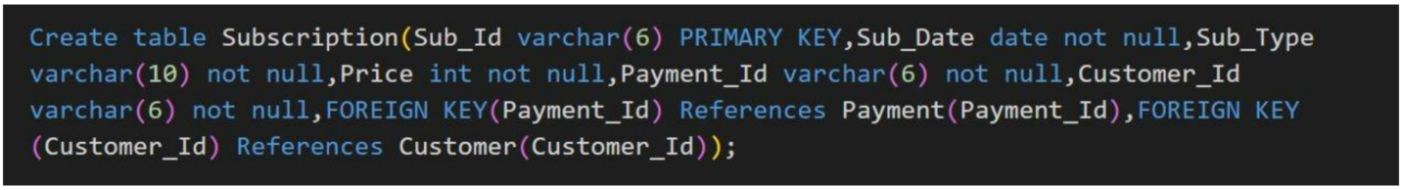
```
Create table Payment(Payment_Id varchar(6) PRIMARY KEY,Payment_Method varchar(15) not
null,Payment_Date Date not null,DueDate Date not null,AmountPaid int not null,
Customer_Id varchar(6) not null);
```

3) Payment History



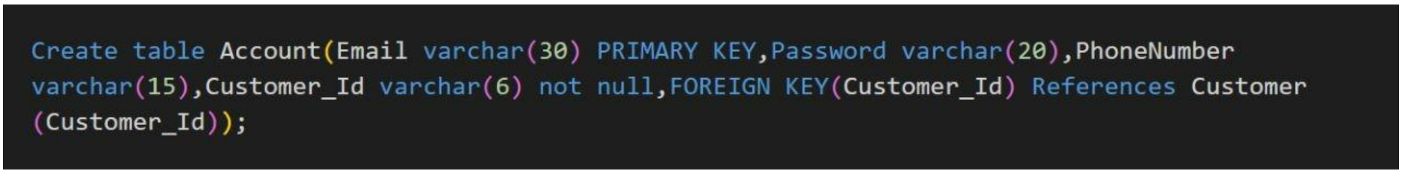
```
Create table Payment_history(History_id varchar(6) PRIMARY KEY,Amount int not null,
startDate date not nul,EndDate date not null,Payment_id varchar(6) not null,FOREIGN KEY
(Payment_id) REFERENCES Payment(Payment_id);
```

4) Subscription



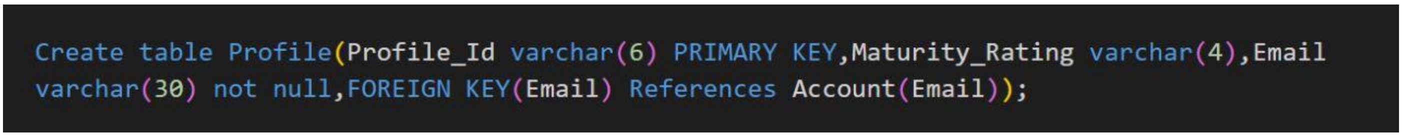
```
Create table Subscription(Sub_Id varchar(6) PRIMARY KEY,Sub_Date date not null,Sub_Type
varchar(10) not null,Price int not null,Payment_Id varchar(6) not null,Customer_Id
varchar(6) not null,FOREIGN KEY(Payment_Id) References Payment(Payment_Id),FOREIGN KEY
(Customer_Id) References Customer(Customer_Id));
```

5) Account



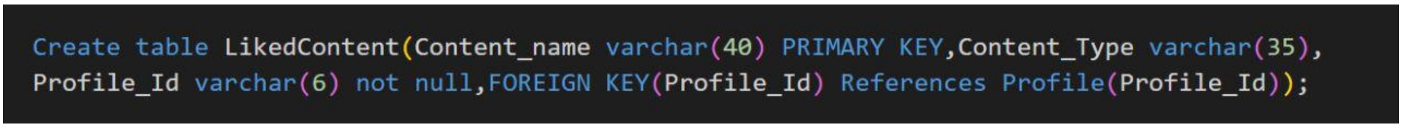
```
Create table Account(Email varchar(30) PRIMARY KEY,Password varchar(20),PhoneNumber
varchar(15),Customer_Id varchar(6) not null,FOREIGN KEY(Customer_Id) References Customer
(Customer_Id));
```

6) Profile



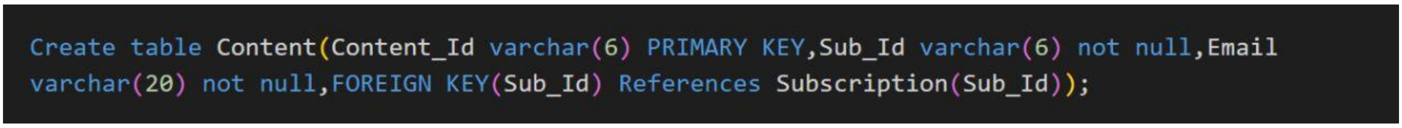
```
Create table Profile(Profile_Id varchar(6) PRIMARY KEY,Maturity_Rating varchar(4),Email
varchar(30) not null,FOREIGN KEY(Email) References Account(Email));
```

7) Liked Content



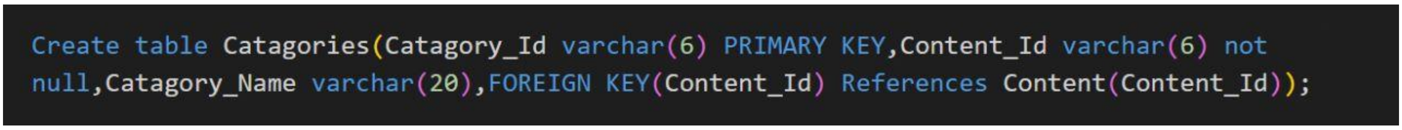
```
Create table LikedContent(Content_name varchar(40) PRIMARY KEY,Content_Type varchar(35),
Profile_Id varchar(6) not null,FOREIGN KEY(Profile_Id) References Profile(Profile_Id));
```

8) Content



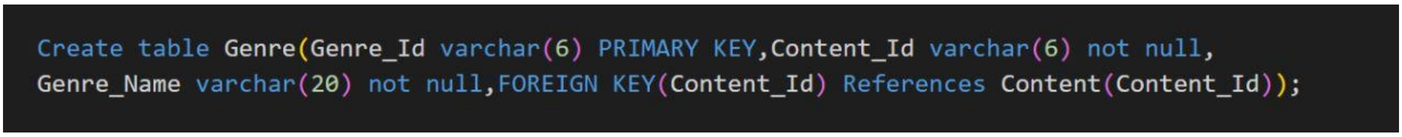
```
Create table Content(Content_Id varchar(6) PRIMARY KEY,Sub_Id varchar(6) not null,Email
varchar(20) not null,FOREIGN KEY(Sub_Id) References Subscription(Sub_Id));
```

9) Categories



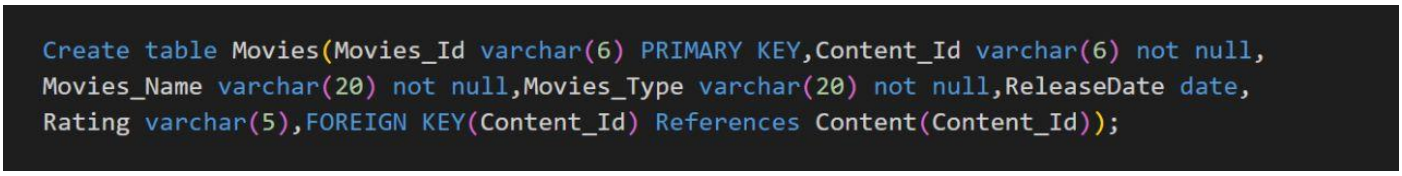
```
Create table Catagories(Catagory_Id varchar(6) PRIMARY KEY,Content_Id varchar(6) not
null,Catagory_Name varchar(20),FOREIGN KEY(Content_Id) References Content(Content_Id));
```

10) Genre



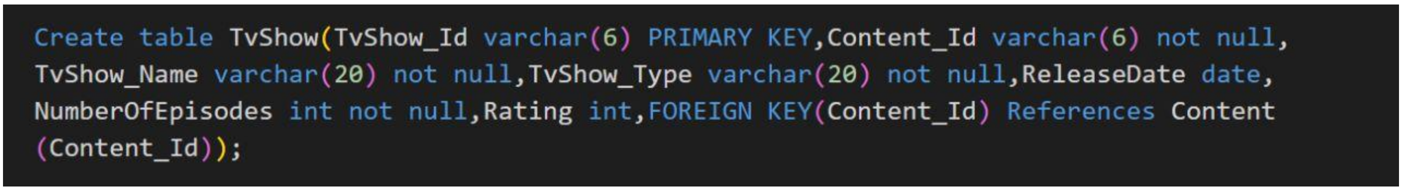
```sql
Create table Genre(Genre_Id varchar(6) PRIMARY KEY,Content_Id varchar(6) not null,
Genre_Name varchar(20) not null,FOREIGN KEY(Content_Id) References Content(Content_Id));
```

11) Movies



```sql
Create table Movies(Movies_Id varchar(6) PRIMARY KEY,Content_Id varchar(6) not null,
Movies_Name varchar(20) not null,Movies_Type varchar(20) not null,ReleaseDate date,
Rating varchar(5),FOREIGN KEY(Content_Id) References Content(Content_Id));
```

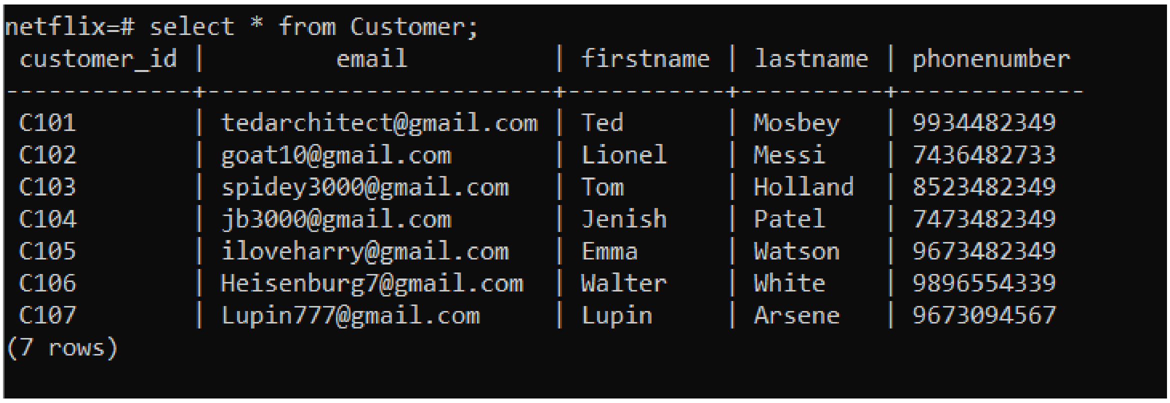
12) TV-Show



```sql
Create table TvShow(TvShow_Id varchar(6) PRIMARY KEY,Content_Id varchar(6) not null,
TvShow_Name varchar(20) not null,TvShow_Type varchar(20) not null,ReleaseDate date,
NumberOfEpisodes int not null,Rating int,FOREIGN KEY(Content_Id) References Content
(Content_Id));
```

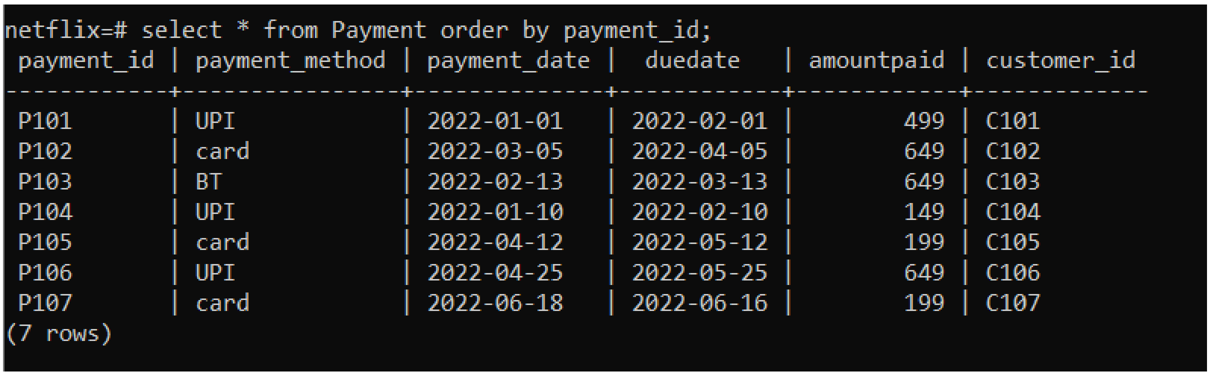
5.2) Inserting Data Values & Creating Tables

1) Customer:-



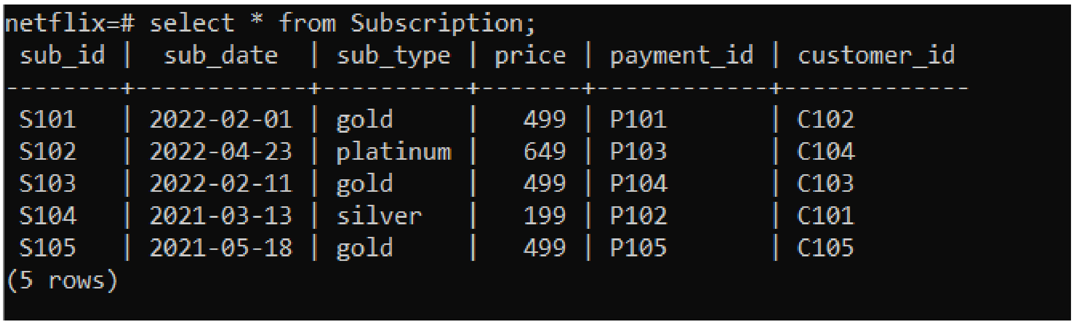
```
netflix=# select * from Customer;
 customer_id |         email          | firstname | lastname | phonenumber
-------------+------------------------+-----------+----------+-------------
 C101        | tedarchitect@gmail.com | Ted       | Mosbey   | 9934482349
 C102        | goat10@gmail.com       | Lionel    | Messi    | 7436482733
 C103        | spidey3000@gmail.com   | Tom       | Holland  | 8523482349
 C104        | jb3000@gmail.com       | Jenish    | Patel    | 7473482349
 C105        | iloveharry@gmail.com   | Emma      | Watson   | 9673482349
 C106        | Heisenburg7@gmail.com  | Walter    | White    | 9896554339
 C107        | Lupin777@gmail.com     | Lupin     | Arsene   | 9673094567
(7 rows)
```

2) Payment:-



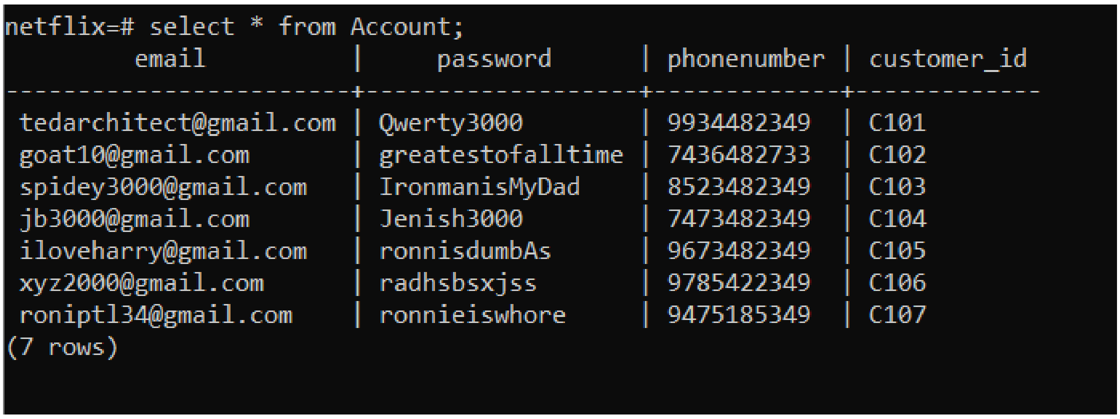
```
netflix=# select * from Payment order by payment_id;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P101       | UPI            | 2022-01-01   | 2022-02-01 |        499 | C101
 P102       | card           | 2022-03-05   | 2022-04-05 |        649 | C102
 P103       | BT             | 2022-02-13   | 2022-03-13 |        649 | C103
 P104       | UPI            | 2022-01-10   | 2022-02-10 |        149 | C104
 P105       | card           | 2022-04-12   | 2022-05-12 |        199 | C105
 P106       | UPI            | 2022-04-25   | 2022-05-25 |        649 | C106
 P107       | card           | 2022-06-18   | 2022-06-16 |        199 | C107
(7 rows)
```

3) Subscription:-



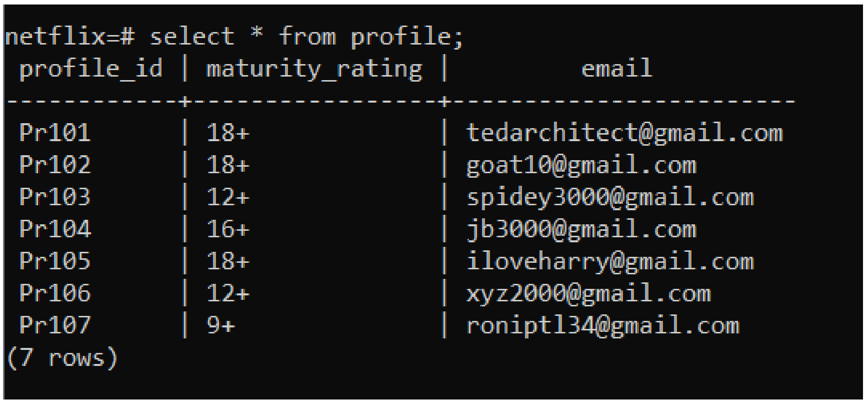
```
netflix=# select * from Subscription;
 sub_id |  sub_date  | sub_type | price | payment_id | customer_id
--------+------------+----------+-------+------------+-------------
 S101   | 2022-02-01 | gold     |   499 | P101       | C102
 S102   | 2022-04-23 | platinum |   649 | P103       | C104
 S103   | 2022-02-11 | gold     |   499 | P104       | C103
 S104   | 2021-03-13 | silver   |   199 | P102       | C101
 S105   | 2021-05-18 | gold     |   499 | P105       | C105
(5 rows)
```

4) **Account**:-



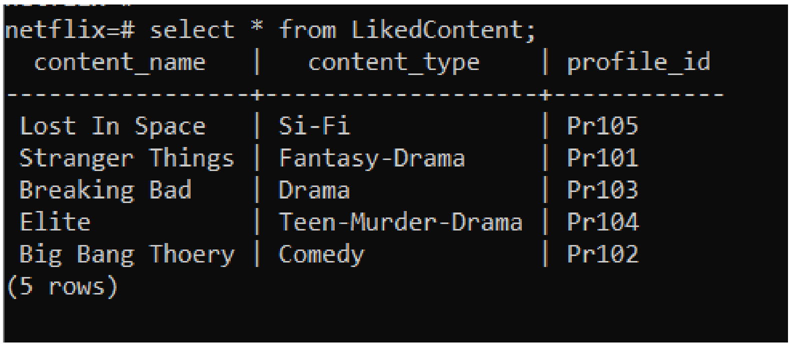
```
netflix=# select * from Account;
        email           |      password      | phonenumber | customer_id
------------------------+--------------------+-------------+-------------
 tedarchitect@gmail.com | Qwerty3000         | 9934482349  | C101
 goat10@gmail.com       | greatestofalltime  | 7436482733  | C102
 spidey3000@gmail.com   | IronmanisMyDad     | 8523482349  | C103
 jb3000@gmail.com       | Jenish3000         | 7473482349  | C104
 iloveharry@gmail.com   | ronnisdumbAs       | 9673482349  | C105
 xyz2000@gmail.com      | radhsbsxjss        | 9785422349  | C106
 roniptl34@gmail.com    | ronnieiswhore      | 9475185349  | C107
(7 rows)
```

5) **Profile**:-



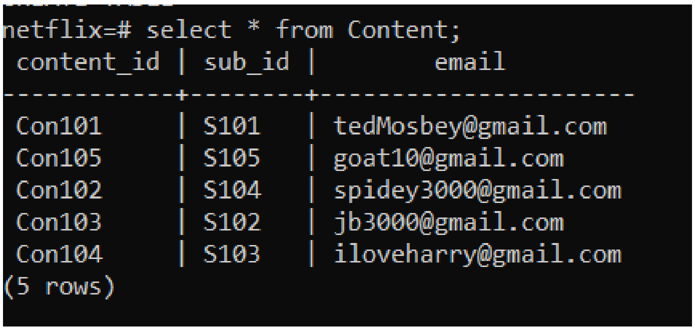
```
netflix=# select * from profile;
 profile_id | maturity_rating |         email
------------+-----------------+------------------------
 Pr101      | 18+             | tedarchitect@gmail.com
 Pr102      | 18+             | goat10@gmail.com
 Pr103      | 12+             | spidey3000@gmail.com
 Pr104      | 16+             | jb3000@gmail.com
 Pr105      | 18+             | iloveharry@gmail.com
 Pr106      | 12+             | xyz2000@gmail.com
 Pr107      | 9+              | roniptl34@gmail.com
(7 rows)
```

6) **LikedContent**:-



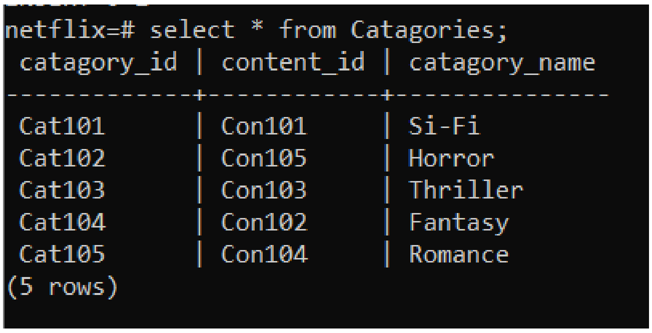
```
netflix=# select * from LikedContent;
  content_name    |    content_type     | profile_id
------------------+---------------------+------------
 Lost In Space    | Si-Fi               | Pr105
 Stranger Things  | Fantasy-Drama       | Pr101
 Breaking Bad     | Drama               | Pr103
 Elite            | Teen-Murder-Drama   | Pr104
 Big Bang Thoery  | Comedy              | Pr102
(5 rows)
```

7) **Content**:-



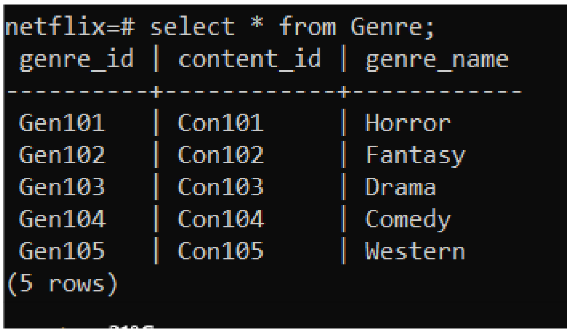
```
netflix=# select * from Content;
 content_id | sub_id |        email
------------+--------+----------------------
 Con101     | S101   | tedMosbey@gmail.com
 Con105     | S105   | goat10@gmail.com
 Con102     | S104   | spidey3000@gmail.com
 Con103     | S102   | jb3000@gmail.com
 Con104     | S103   | iloveharry@gmail.com
(5 rows)
```

8) **Categories**:-



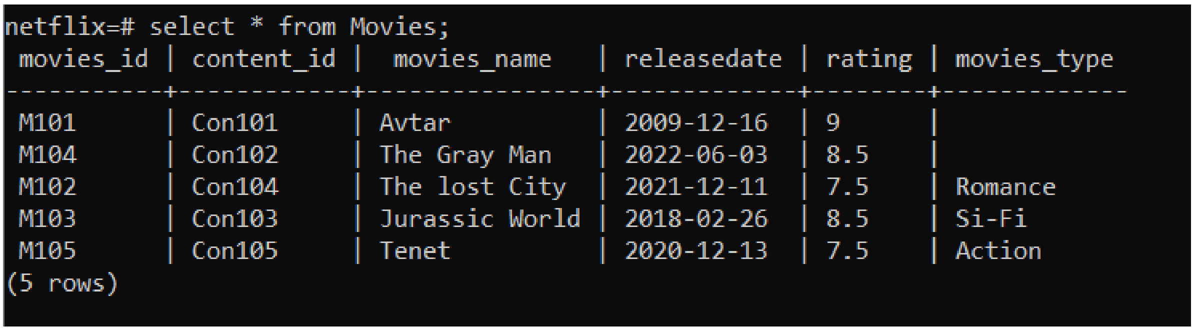
```
netflix=# select * from Catagories;
 catagory_id | content_id | catagory_name
-------------+------------+---------------
 Cat101      | Con101     | Si-Fi
 Cat102      | Con105     | Horror
 Cat103      | Con103     | Thriller
 Cat104      | Con102     | Fantasy
 Cat105      | Con104     | Romance
(5 rows)
```

9) **Genre**:-



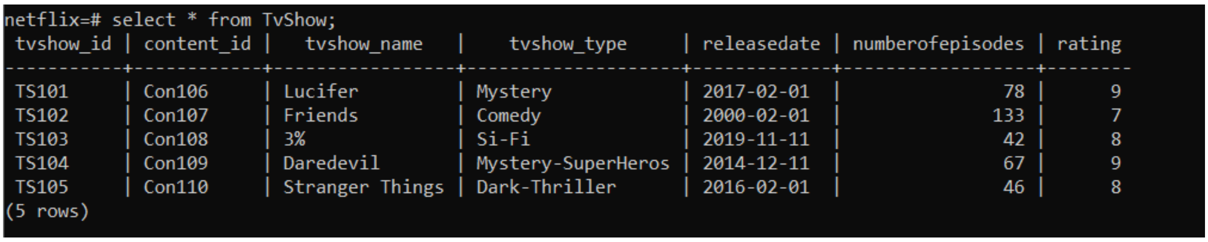
```
netflix=# select * from Genre;
 genre_id | content_id | genre_name
----------+------------+------------
 Gen101   | Con101     | Horror
 Gen102   | Con102     | Fantasy
 Gen103   | Con103     | Drama
 Gen104   | Con104     | Comedy
 Gen105   | Con105     | Western
(5 rows)
```

10) **Movies:-**



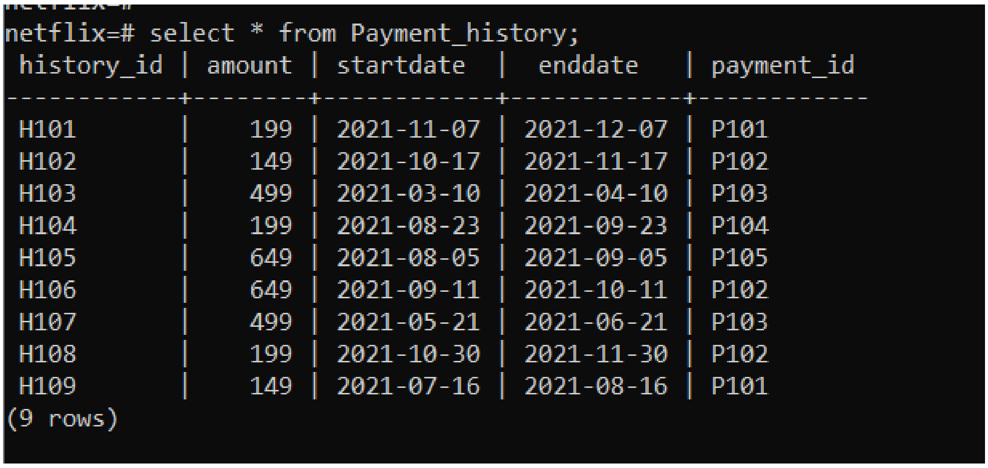
```
netflix=# select * from Movies;
 movies_id | content_id |  movies_name   | releasedate | rating | movies_type
-----------+------------+----------------+-------------+--------+-------------
 M101      | Con101     | Avtar          | 2009-12-16  | 9      |
 M104      | Con102     | The Gray Man   | 2022-06-03  | 8.5    |
 M102      | Con104     | The lost City  | 2021-12-11  | 7.5    | Romance
 M103      | Con103     | Jurassic World | 2018-02-26  | 8.5    | Si-Fi
 M105      | Con105     | Tenet          | 2020-12-13  | 7.5    | Action
(5 rows)
```

11) Tvshow:-



```
netflix=# select * from TvShow;
 tvshow_id | content_id |   tvshow_name    |     tvshow_type      | releasedate | numberofepisodes | rating
-----------+------------+------------------+----------------------+-------------+------------------+--------
 TS101     | Con106     | Lucifer          | Mystery              | 2017-02-01  |               78 |      9
 TS102     | Con107     | Friends          | Comedy               | 2000-02-01  |              133 |      7
 TS103     | Con108     | 3%               | Si-Fi                | 2019-11-11  |               42 |      8
 TS104     | Con109     | Daredevil        | Mystery-SuperHeros   | 2014-12-11  |               67 |      9
 TS105     | Con110     | Stranger Things  | Dark-Thriller        | 2016-02-01  |               46 |      8
(5 rows)
```

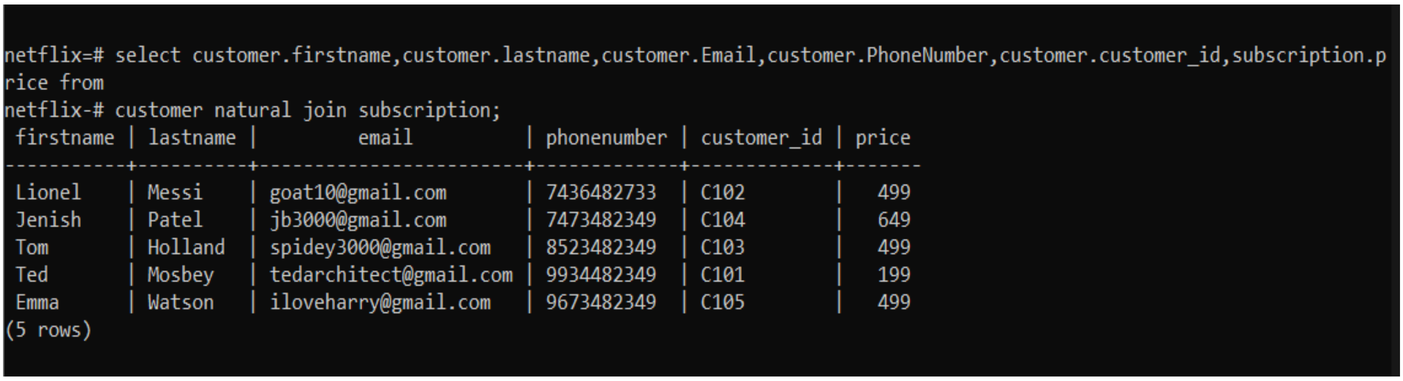
12) Payment History:-



```
netflix=# select * from Payment_history;
 history_id | amount | startdate  |  enddate   | payment_id
------------+--------+------------+------------+------------
 H101       |    199 | 2021-11-07 | 2021-12-07 | P101
 H102       |    149 | 2021-10-17 | 2021-11-17 | P102
 H103       |    499 | 2021-03-10 | 2021-04-10 | P103
 H104       |    199 | 2021-08-23 | 2021-09-23 | P104
 H105       |    649 | 2021-08-05 | 2021-09-05 | P105
 H106       |    649 | 2021-09-11 | 2021-10-11 | P102
 H107       |    499 | 2021-05-21 | 2021-06-21 | P103
 H108       |    199 | 2021-10-30 | 2021-11-30 | P102
 H109       |    149 | 2021-07-16 | 2021-08-16 | P101
(9 rows)
```

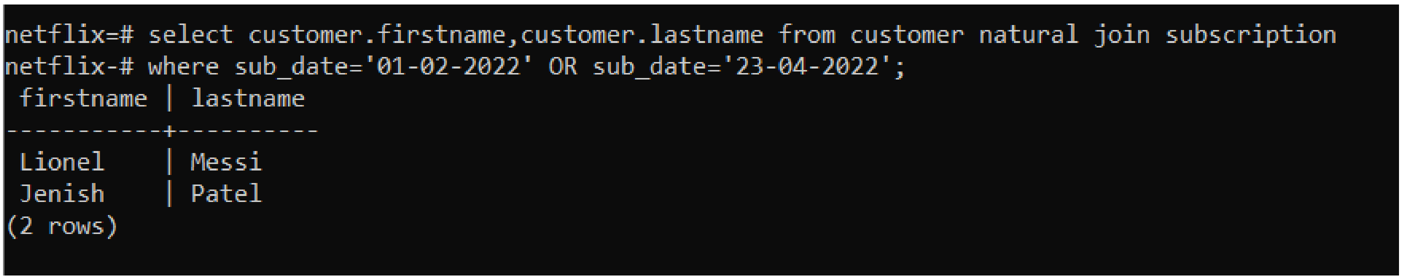
5.3) Queries using Basic DBMS Constraints , Joins & Subqueries.

1) Display Customers details who have taken a Subscription with their Subscription plan price.



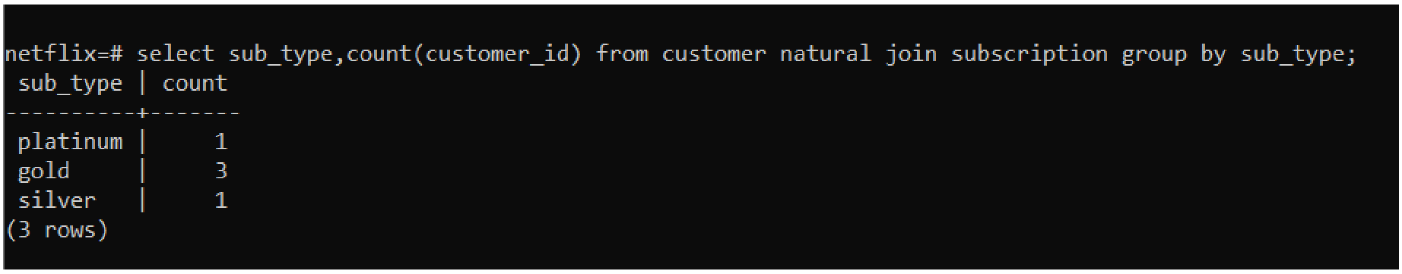
```
netflix=# select customer.firstname,customer.lastname,customer.Email,customer.PhoneNumber,customer.customer_id,subscription.p
rice from
netflix-# customer natural join subscription;
 firstname | lastname |         email          | phonenumber | customer_id | price
-----------+----------+------------------------+-------------+-------------+-------
 Lionel    | Messi    | goat10@gmail.com       | 7436482733  | C102        |   499
 Jenish    | Patel    | jb3000@gmail.com       | 7473482349  | C104        |   649
 Tom       | Holland  | spidey3000@gmail.com   | 8523482349  | C103        |   499
 Ted       | Mosbey   | tedarchitect@gmail.com | 9934482349  | C101        |   199
 Emma      | Watson   | iloveharry@gmail.com   | 9673482349  | C105        |   499
(5 rows)
```

2) Find the number of customers who have done payment of subscription on specific date and display their names.



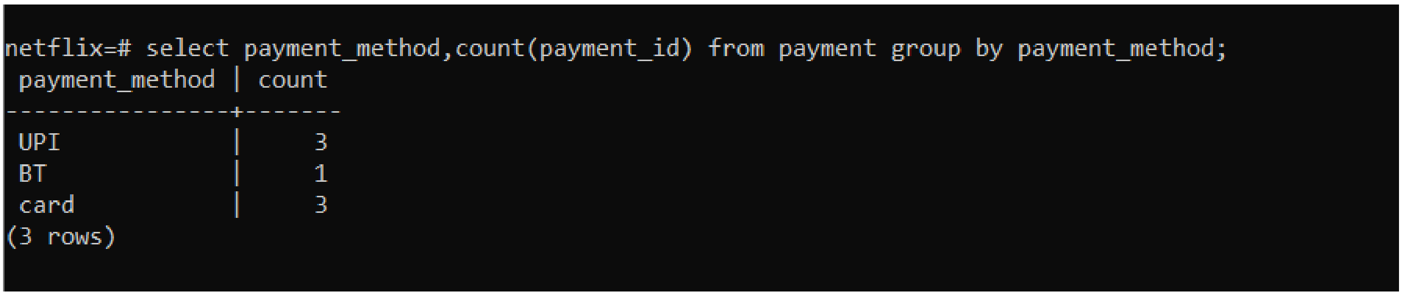
```
netflix=# select customer.firstname,customer.lastname from customer natural join subscription
netflix-# where sub_date='01-02-2022' OR sub_date='23-04-2022';
 firstname | lastname
-----------+----------
 Lionel    | Messi
 Jenish    | Patel
(2 rows)
```

3) Display the number of customers for a specific subscription plan type.



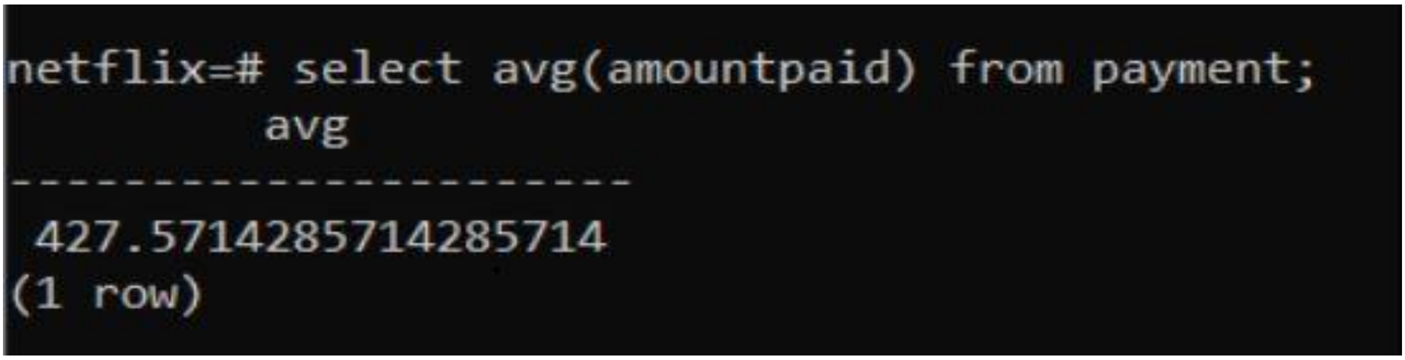
```
netflix=# select sub_type,count(customer_id) from customer natural join subscription group by sub_type;
 sub_type | count
----------+-------
 platinum |     1
 gold     |     3
 silver   |     1
(3 rows)
```

4) Display the count of persons who has done payment with same payment Method.



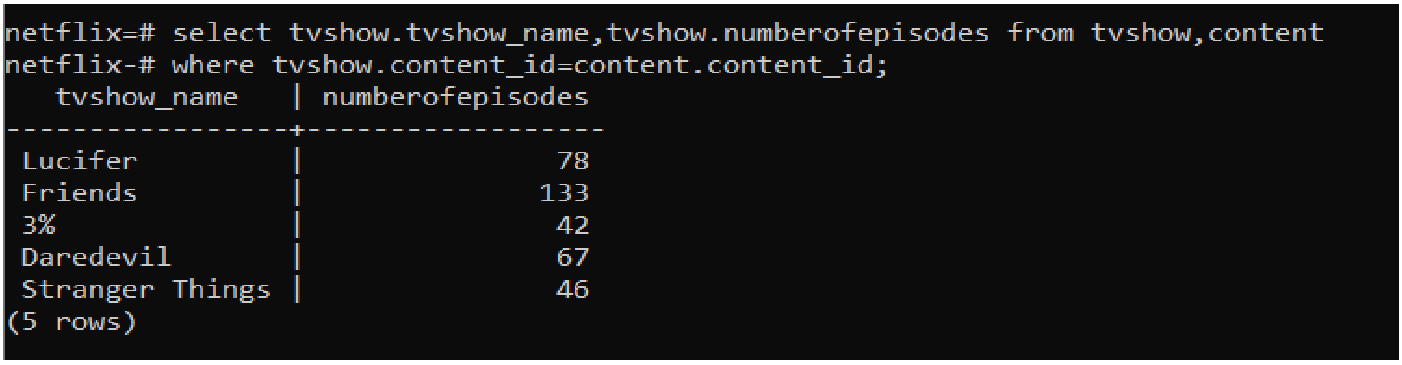
```
netflix=# select payment_method,count(payment_id) from payment group by payment_method;
 payment_method | count
----------------+-------
 UPI            |     3
 BT             |     1
 card           |     3
(3 rows)
```

5) Display the Average amount of payment done by customers.



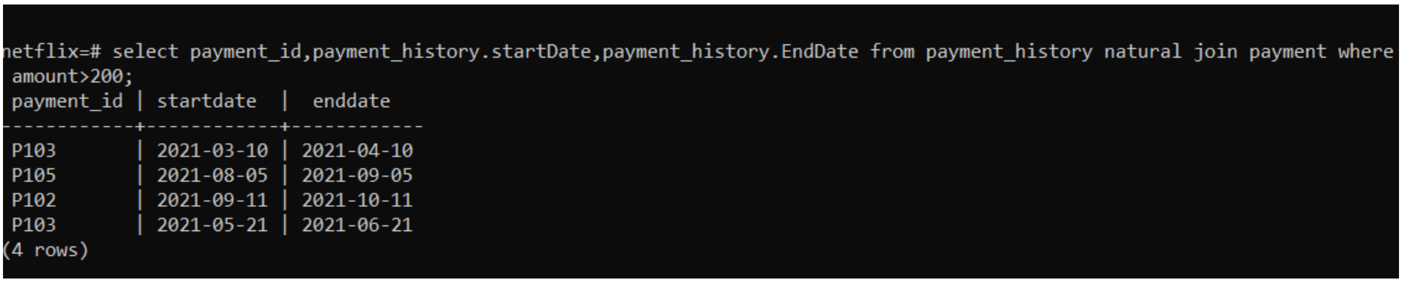
```
netflix=# select avg(amountpaid) from payment;
         avg
---------------------
 427.5714285714285714
(1 row)
```

6) Display total number of episodes according to the TVShow.



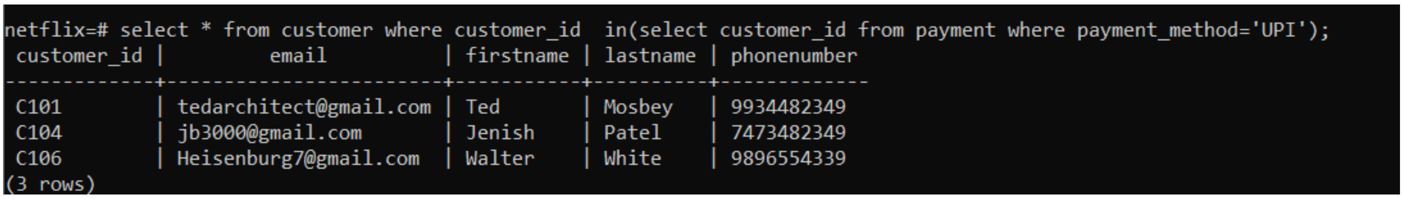
```
netflix=# select tvshow.tvshow_name,tvshow.numberofepisodes from tvshow,content
netflix-# where tvshow.content_id=content.content_id;
   tvshow_name    | numberofepisodes
------------------+------------------
 Lucifer          |               78
 Friends          |              133
 3%               |               42
 Daredevil        |               67
 Stranger Things  |               46
(5 rows)
```

7) Display Start Date & End date of a Subscription plan where payment is done in past with same payment Id, who has paid more than 200.



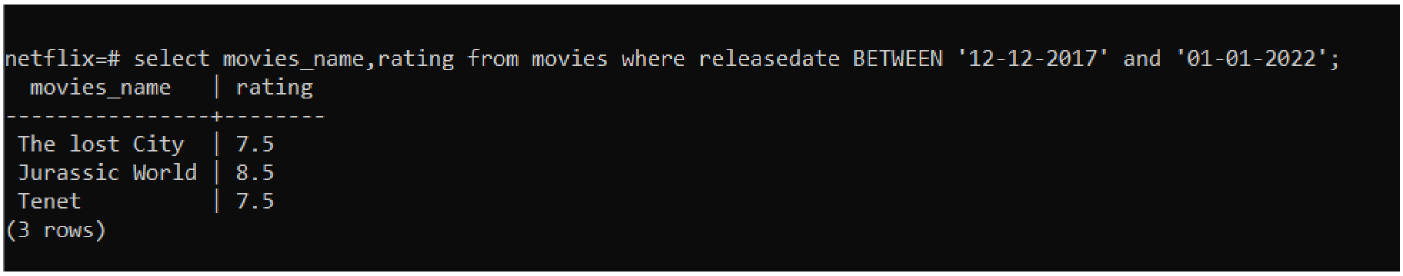
```
etflix=# select payment_id,payment_history.startDate,payment_history.EndDate from payment_history natural join payment where
 amount>200;
 payment_id | startdate  |  enddate
------------+------------+------------
 P103       | 2021-03-10 | 2021-04-10
 P105       | 2021-08-05 | 2021-09-05
 P102       | 2021-09-11 | 2021-10-11
 P103       | 2021-05-21 | 2021-06-21
(4 rows)
```

8) Display details of a Customers who has done payment with ‘UPI’.



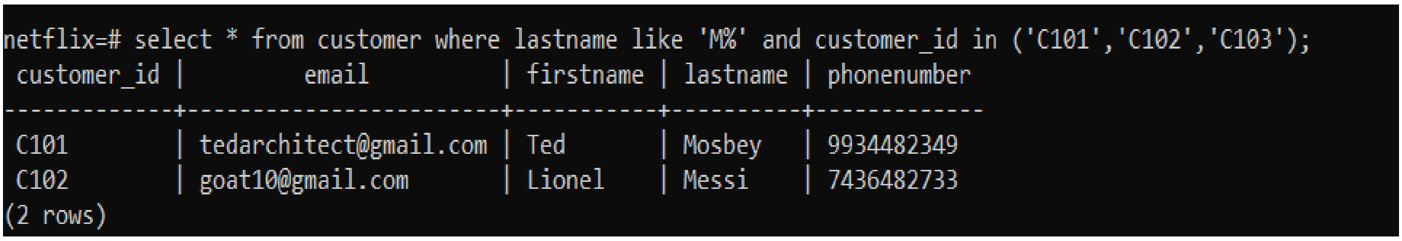
```
netflix=# select * from customer where customer_id  in(select customer_id from payment where payment_method='UPI');
 customer_id |         email          | firstname | lastname | phonenumber
-------------+------------------------+-----------+----------+-------------
 C101        | tedarchitect@gmail.com | Ted       | Mosbey   | 9934482349
 C104        | jb3000@gmail.com       | Jenish    | Patel    | 7473482349
 C106        | Heisenburg7@gmail.com  | Walter    | White    | 9896554339
(3 rows)
```

9) Display movies Name & Rating of a movies which is released after 2017 & Before 2022 and Has a movie type.



```
netflix=# select movies_name,rating from movies where releasedate BETWEEN '12-12-2017' and '01-01-2022';
  movies_name   | rating
----------------+--------
 The lost City  | 7.5
 Jurassic World | 8.5
 Tenet          | 7.5
(3 rows)
```

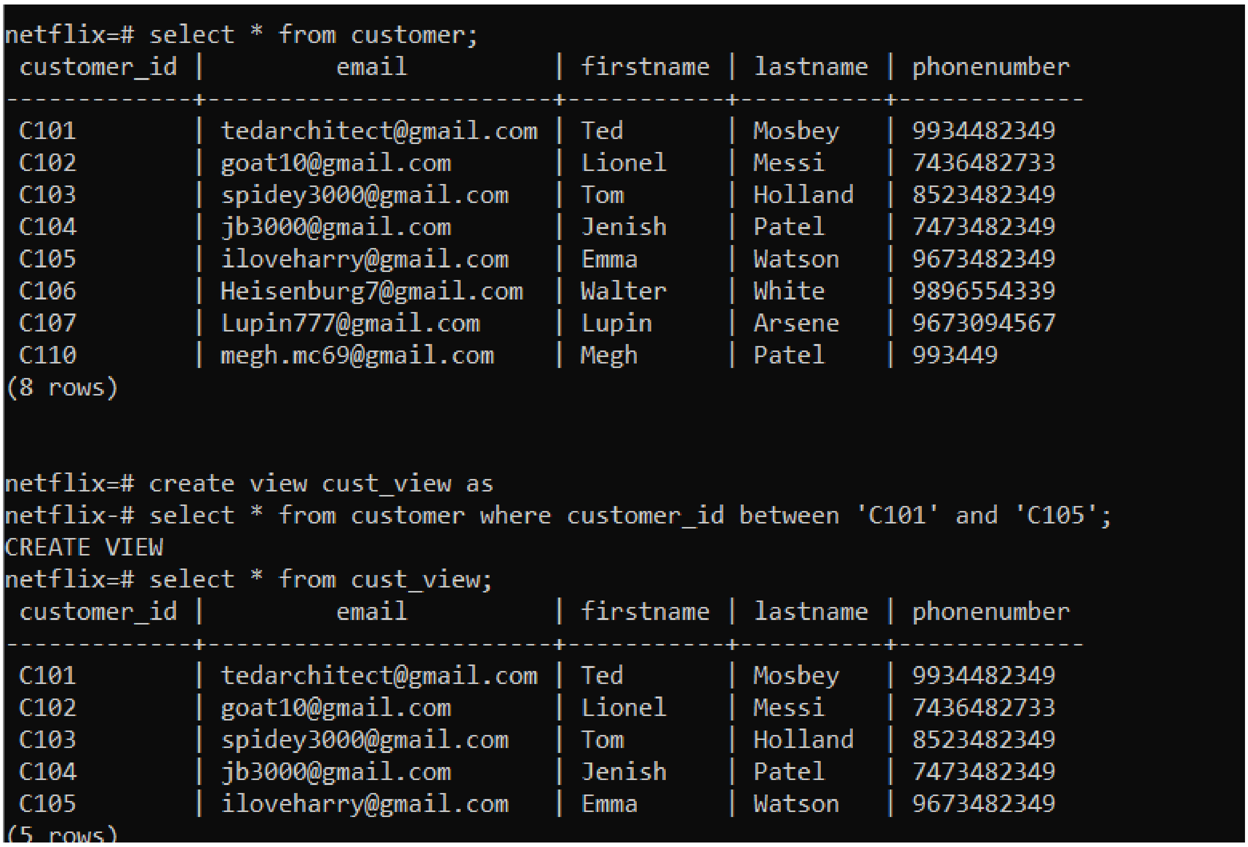
10) Display Customers Details who has last name starting with ‘M’ & customer Id as C101,C102,C103.



```
netflix=# select * from customer where lastname like 'M%' and customer_id in ('C101','C102','C103');
 customer_id |         email          | firstname | lastname | phonenumber
-------------+------------------------+-----------+----------+-------------
 C101        | tedarchitect@gmail.com | Ted       | Mosbey   | 9934482349
 C102        | goat10@gmail.com       | Lionel    | Messi    | 7436482733
(2 rows)
```

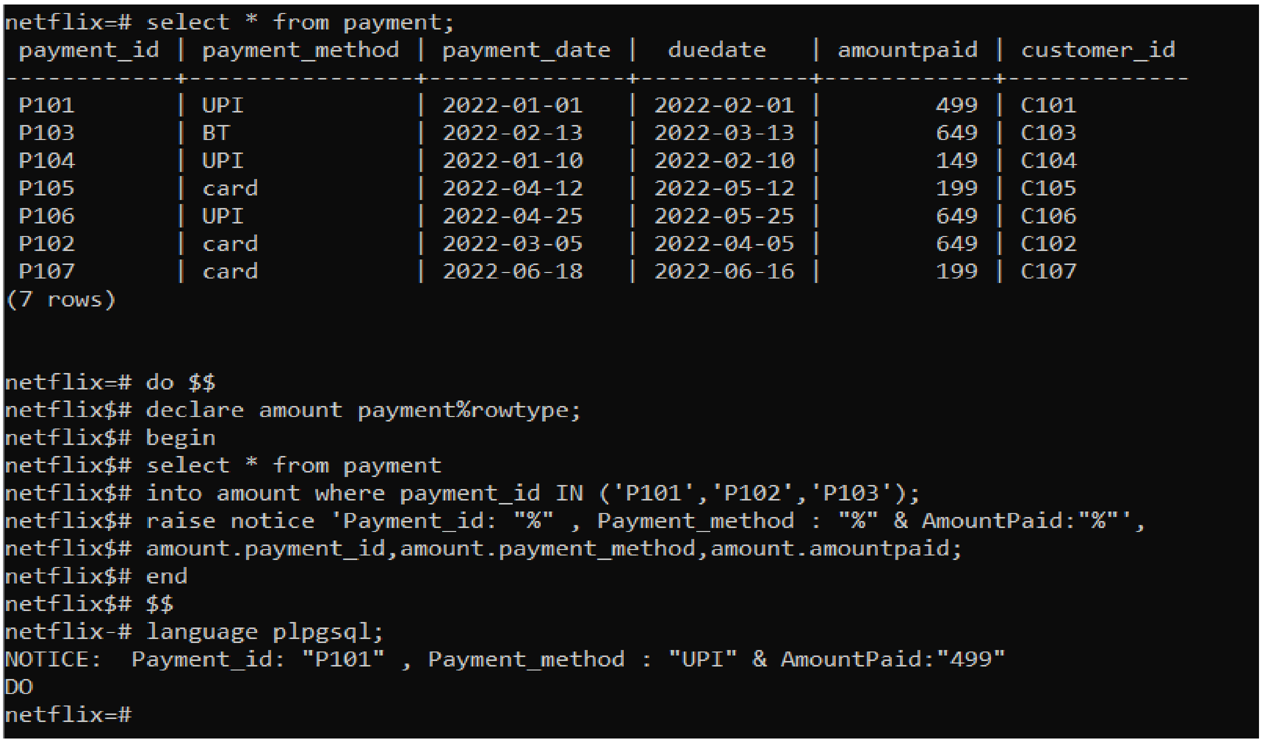
5.4) PL/SQL Blocks(Views)

1. View



```
netflix=# select * from customer;
 customer_id |         email          | firstname | lastname | phonenumber
-------------+------------------------+-----------+----------+-------------
 C101        | tedarchitect@gmail.com | Ted       | Mosbey   | 9934482349
 C102        | goat10@gmail.com       | Lionel    | Messi    | 7436482733
 C103        | spidey3000@gmail.com   | Tom       | Holland  | 8523482349
 C104        | jb3000@gmail.com       | Jenish    | Patel    | 7473482349
 C105        | iloveharry@gmail.com   | Emma      | Watson   | 9673482349
 C106        | Heisenburg7@gmail.com  | Walter    | White    | 9896554339
 C107        | Lupin777@gmail.com     | Lupin     | Arsene   | 9673094567
 C110        | megh.mc69@gmail.com    | Megh      | Patel    | 993449
(8 rows)


netflix=# create view cust_view as
netflix-# select * from customer where customer_id between 'C101' and 'C105';
CREATE VIEW
netflix=# select * from cust_view;
 customer_id |         email          | firstname | lastname | phonenumber
-------------+------------------------+-----------+----------+-------------
 C101        | tedarchitect@gmail.com | Ted       | Mosbey   | 9934482349
 C102        | goat10@gmail.com       | Lionel    | Messi    | 7436482733
 C103        | spidey3000@gmail.com   | Tom       | Holland  | 8523482349
 C104        | jb3000@gmail.com       | Jenish    | Patel    | 7473482349
 C105        | iloveharry@gmail.com   | Emma      | Watson   | 9673482349
(5 rows)
```

2. RowType



```
netflix=# select * from payment;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P101       | UPI            | 2022-01-01   | 2022-02-01 |        499 | C101
 P103       | BT             | 2022-02-13   | 2022-03-13 |        649 | C103
 P104       | UPI            | 2022-01-10   | 2022-02-10 |        149 | C104
 P105       | card           | 2022-04-12   | 2022-05-12 |        199 | C105
 P106       | UPI            | 2022-04-25   | 2022-05-25 |        649 | C106
 P102       | card           | 2022-03-05   | 2022-04-05 |        649 | C102
 P107       | card           | 2022-06-18   | 2022-06-16 |        199 | C107
(7 rows)


netflix=# do $$
netflix$# declare amount payment%rowtype;
netflix$# begin
netflix$# select * from payment
netflix$# into amount where payment_id IN ('P101','P102','P103');
netflix$# raise notice 'Payment_id: "%" , Payment_method : "%" & AmountPaid:"%"',
netflix$# amount.payment_id,amount.payment_method,amount.amountpaid;
netflix$# end
netflix$# $$
netflix-# language plpgsql;
NOTICE:  Payment_id: "P101" , Payment_method : "UPI" & AmountPaid:"499"
DO
netflix=#
```

s

5.5) Functions & Triggers

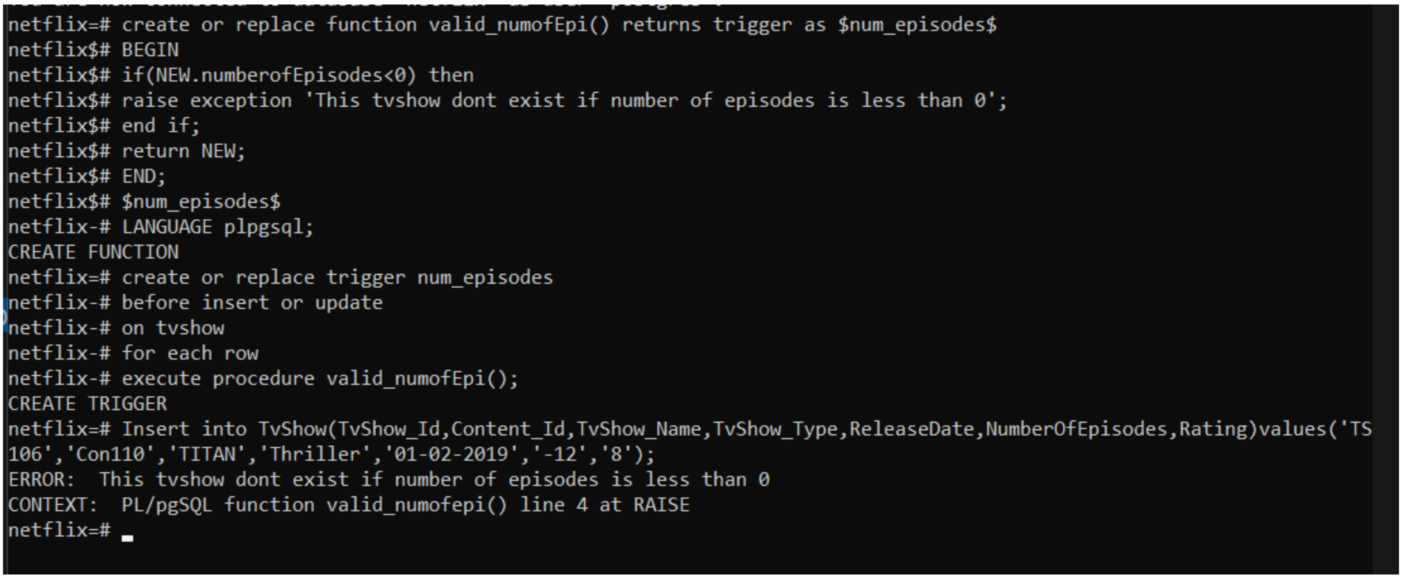
1.) Create a function & Trigger for invalid entry Of a total numbers of episodes in a TV-Show.

FUNCTION:-

```
create or replace function valid_numofEpi() returns trigger as $num_episodes$
BEGIN
if(NEW.numberofEpisodes<0) then
    raise exception 'This tvshowdont exist if number of episodes is less than 0';
end if;
return NEW;
END;
$num_episodes$
LANGUAGE plpgsql;
```

TRIGGER:

```
create or replace trigger num_episodes
before insert or update
on tvshow
for each row
execute procedure valid_numofEpi();
```



```
netflix=# create or replace function valid_numofEpi() returns trigger as $num_episodes$
netflix$# BEGIN
netflix$# if(NEW.numberofEpisodes<0) then
netflix$# raise exception 'This tvshow dont exist if number of episodes is less than 0';
netflix$# end if;
netflix$# return NEW;
netflix$# END;
netflix$# $num_episodes$
netflix-# LANGUAGE plpgsql;
CREATE FUNCTION
netflix=# create or replace trigger num_episodes
netflix-# before insert or update
netflix-# on tvshow
netflix-# for each row
netflix-# execute procedure valid_numofEpi();
CREATE TRIGGER
netflix=# Insert into TvShow(TvShow_Id,Content_Id,TvShow_Name,TvShow_Type,ReleaseDate,NumberOfEpisodes,Rating)values('TS
106','Con110','TITAN','Thriller','01-02-2019','-12','8');
ERROR:  This tvshow dont exist if number of episodes is less than 0
CONTEXT:  PL/pgSQL function valid_numofepi() line 4 at RAISE
netflix=# _
```

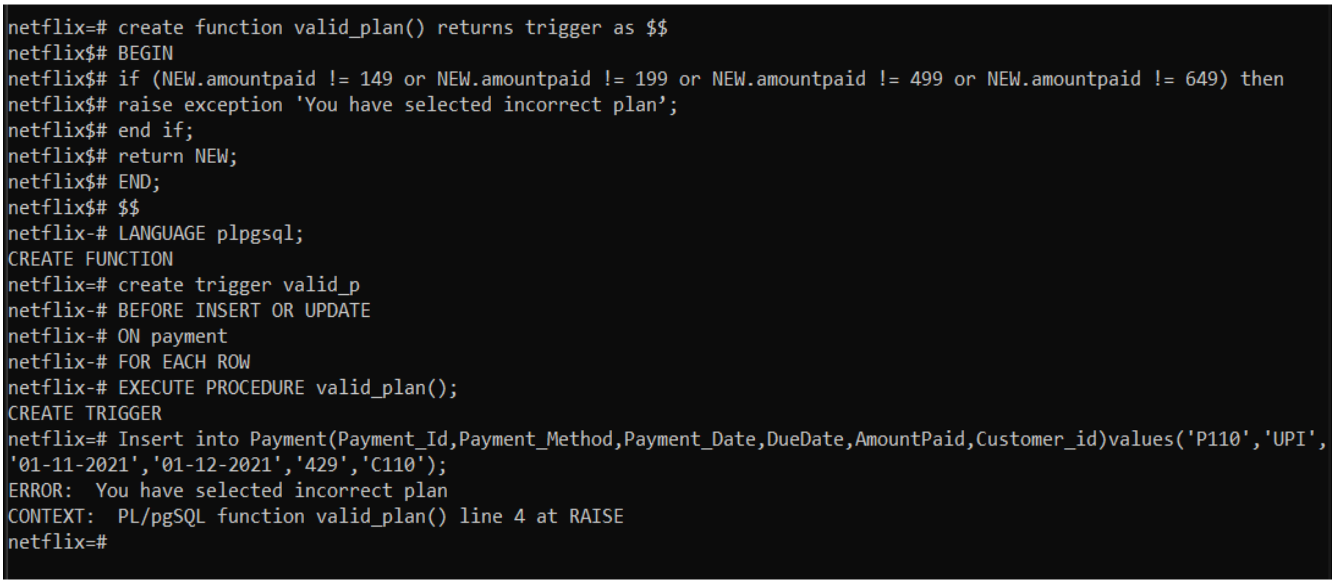
2.) Create a function & Trigger for user make Payment of Invalid amount like 300 because there are no plan with this amount.

FUNCTION:-

```
create function valid_plan() returns trigger as
BEGIN
if (NEW.amountpaid != 149 or NEW.amountpaid != 199 or NEW.amountpaid != 499 or
NEW.amountpaid != 649) then
raise exception 'You have selected incorrect plan’;
end if;
return NEW;
END;
$$
LANGUAGE plpgsql;
```

TRIGGER:-

```
create trigger valid_p
BEFORE INSERT OR UPDATE
ON payment
FOR EACH ROW
EXECUTE PROCEDURE valid_plan();
```



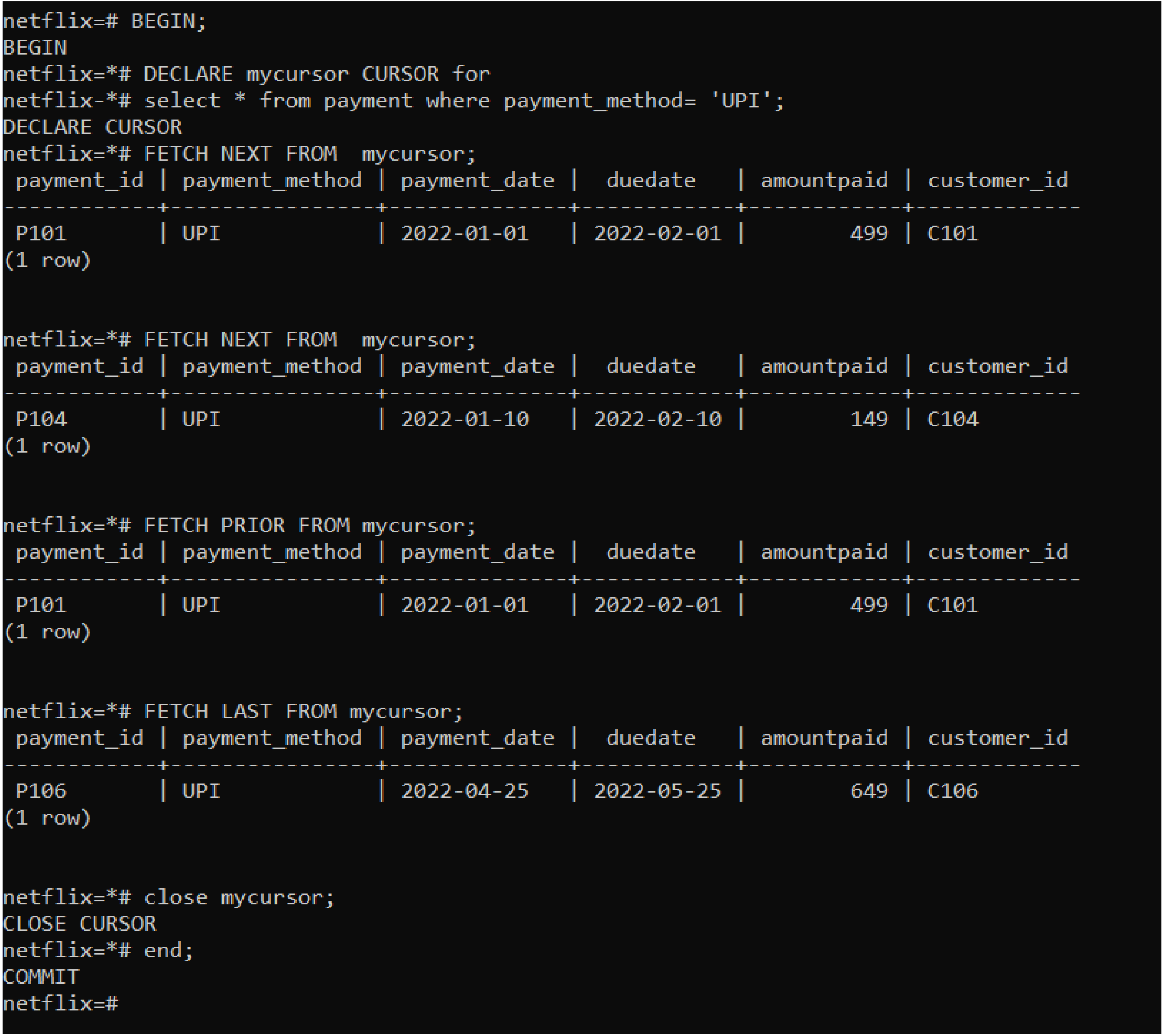
```
netflix=# create function valid_plan() returns trigger as $$
netflix$# BEGIN
netflix$# if (NEW.amountpaid != 149 or NEW.amountpaid != 199 or NEW.amountpaid != 499 or NEW.amountpaid != 649) then
netflix$# raise exception 'You have selected incorrect plan’;
netflix$# end if;
netflix$# return NEW;
netflix$# END;
netflix$# $$
netflix-# LANGUAGE plpgsql;
CREATE FUNCTION
netflix=# create trigger valid_p
netflix-# BEFORE INSERT OR UPDATE
netflix-# ON payment
netflix-# FOR EACH ROW
netflix-# EXECUTE PROCEDURE valid_plan();
CREATE TRIGGER
netflix=# Insert into Payment(Payment_Id,Payment_Method,Payment_Date,DueDate,AmountPaid,Customer_id)values('P110','UPI',
'01-11-2021','01-12-2021','429','C110');
ERROR:  You have selected incorrect plan
CONTEXT:  PL/pgSQL function valid_plan() line 4 at RAISE
netflix=#
```

5.6) Cursor:-

1. Create a Cursor which traverses through a payment table where & payment method is ‘UPI’.

CURSOR:-

BEGIN;
DECLARE mycursor CURSOR for
select * from payment where payment_method= 'UPI';
FETCH NEXT FROM mycursor;
FETCH PRIOR FROM mycursor;
FETCH LAST FROM mycursor;
CLOSE mycursor;
end;



```
netflix=# BEGIN;
BEGIN
netflix=*# DECLARE mycursor CURSOR for
netflix-*# select * from payment where payment_method= 'UPI';
DECLARE CURSOR
netflix=*# FETCH NEXT FROM  mycursor;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P101       | UPI            | 2022-01-01   | 2022-02-01 |        499 | C101
(1 row)


netflix=*# FETCH NEXT FROM  mycursor;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P104       | UPI            | 2022-01-10   | 2022-02-10 |        149 | C104
(1 row)


netflix=*# FETCH PRIOR FROM mycursor;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P101       | UPI            | 2022-01-01   | 2022-02-01 |        499 | C101
(1 row)


netflix=*# FETCH LAST FROM mycursor;
 payment_id | payment_method | payment_date |  duedate   | amountpaid | customer_id
------------+----------------+--------------+------------+------------+-------------
 P106       | UPI            | 2022-04-25   | 2022-05-25 |        649 | C106
(1 row)


netflix=*# close mycursor;
CLOSE CURSOR
netflix=*# end;
COMMIT
netflix=#
```

6.) FUTURE ENHANCEMENTS OF THE SYSTEM

- We will design Front-end using React Framework and Develop Back-end in NodeJS.
- Methods and user data input will be a lot easy after the implementation of GUI.
- In the future, we can place the system on the cloud so the maintenance of the data can be reduced.

7. BIBLIOGRAPHY

- We created ER-Model on Whimsical and Relational Schema on MySQL WorkBench.
- ER-MODEL - https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD

- For the implementation of this project, we referred to materials shared by Prof. Archana N. Vyas and the following websites and books:

Book:

Database System Concepts

-Henry F. Korth & A. Silberschatz 2nd Ed. McGraw-Hill 1991

Websites:

- https://www.w3schools.com/sql/sql_syntax.asp
- https://www.tutorialspoint.com/
- https://dev.mysql.com/doc/
- https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/